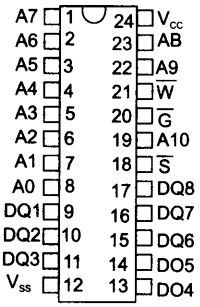
Figure 9–4 illustrates the 4016 SRAM, which is a 2K × 8 read/write memory. This device has 11 address inputs and eight data input/output connections. This device is representative of all SRAM devices, except for the number of address and data connections.

The control inputs of this RAM are slightly different from those presented earlier. The OE pin is labeled G, the CS pin is S, and the WE pin is W. Despite the altered designations, the control pins function exactly the same as those outlined previously. Other manufacturers make this popular SRAM under the part numbers 2016 and 6116.

Figure 9–5 depicts the timing diagram for the 4016 SRAM. As the read cycle timing reveals, the access time is t_a (A). On the slowest version of the 4016, this time is 250 ns, which is fast enough to connect directly to an 8088 or an 8086 operated at 5 MHz without wait states. Again, it is important to remember that the access time must be checked to determine the compatibility of memory components with the microprocessor.

Figure 9–6 illustrates the pin-out of the 62256, 32K × 8 static RAM. This device is packaged in a 28-pin integrated circuit, and is available with access times of 120 ns or 150 ns. Other common SRAM devices are available in 8K × 8, 128K × 8, and 256K × 8 sizes, with access times of as little as 10 ns for SRAM used in computer cache memory systems.



PIN NOMENCLA TURE	
A0 – A10	Addresses
DQ1 – DQ8	Data In/Data Out
$\overline{G}$	Output Enable
$\overline{S}$	Chip Select
V_{cc}	+ 5-6 Supply
V_{ss}	Ground
$\overline{W}$	Write Enable

FIGURE 9–4 The pin-out of the TMS4016, 2K × 8 static RAM (SRAM). (Courtesy of Texas Instruments Incorporated.)

Dynamic RAM (DRAM) Memory

About the largest static RAM available today is a 128K × 8. Dynamic RAM, on the other hand, is available in much larger sizes: up to 64M × 1. In all other respects, DRAM is essentially the same as SRAM, except that it retains data for only 2 or 4 ms on an integrated capacitor. After 2 or 4 ms, the contents of the DRAM must be completely rewritten *(refreshed)* because the capacitors, which store a logic 1 or logic 0, lose their charges.

Instead of requiring the almost impossible task of reading the contents of each memory location with a program and then rewriting them, the manufacturer has internally constructed the DRAM differently from the SRAM. In the DRAM, the entire contents of the memory is refreshed with 256 reads in a 2- or 4-ms interval. Refreshing also occurs during a write, a read, or during a special refresh cycle.

Another disadvantage of DRAM memory is that it requires so many address pins that the manufacturers have decided to multiplex the address inputs. Figure 9–7 illustrates a 64K × 4 DRAM, the TMS4464, which stores 256K bits of data. Notice that it contains only eight address inputs where it should contain 16—the number required to address 64K memory locations. The only way that 16 address bits can be forced into eight address pins is in two 8-bit increments. This operation requires two special pins: the **column address strobe** ($\overline{CAS}$) and **row address strobe** ($\overline{RAS}$). First, A0–A7 are placed on the address pins and strobed into an internal row latch by $\overline{RAS}$ as the row address. Next, the address bits A8–A15 are placed on the same eight address inputs and strobed into an internal column latch by $\overline{CAS}$ as the column address (see Figure 9–8 for this timing). The 16-bit address held in these internal latches addresses the contents of one of the 4-bit memory locations. Note that $\overline{CAS}$ also performs the function of the chip selection input to the DRAM.

Figure 9–9 illustrates a set of multiplexers used to strobe the column and row addresses into the eight address pins on a pair of TMS4464 DRAMs. Here, the $\overline{RAS}$ signal not only strobes the row address into the DRAMs, but it also selects which part of the address is applied to the address inputs. This is possible due to the long propagation-delay time of the multiplexers. When is RAS a logic 1, the B inputs are connected to the Y outputs of the multiplexers; when the RAS input goes to a logic 0, the A inputs connect to the Y outputs. Because the internal row address latch is edge-triggered, it captures the row address before the address at the inputs changes to the column address. More detail on DRAM and DRAM interfacing is provided in Section 9–6.

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER		TEST CONDITIONS		MIN	TYP†	MAX	UNIT
V_{OH}	High level voltage	I_{OH} = -1 mA,	V_{CC} = 4.5 V	2.4			V
V_{OL}	Low level voltage	I_{OL} = 2.1 mA,	V_{CC} = 4.5 V			0.4	V
I_I	Input current	V_I = 0 V to 5.5 V				10	μA
I_{OZ}	Off-state output current	$\overline{S}$ or $\overline{G}$ at 2 V or $\overline{W}$ at 0.8 V, V_O = 0 V to 5.5 V				10	μA
I_{CC}	Supply current from V_{CC}	I_O = 0 mA, T_A = 0°C (worst case)	V_{CC} = 5.5 V,		40	70	mA
C_i	Input capacitance	V_I = 0 V,	f = 1 MHz			8	pF
C_o	Output capacitance	V_O = 0 V,	f = 1 MHz			12	pF

†All typical values are at V_{CC} = 5 V, T_A = 25°C.

timing requirements over recommended supply voltage range and operating free-air temperature range

PARAMETER		TMS4016-12		TMS4016-15		TMS4016-20		TMS4016-25		UNIT
		MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	
$t_{c(rd)}$	Read cycle time	120		150		200		250		ns
$t_{c(wr)}$	Write cycle time	120		150		200		250		ns
$t_{w(W)}$	Write pulse width	60		80		100		120		ns
$t_{su(A)}$	Address setup time	20		20		20		20		ns
$t_{su(S)}$	Chip select setup time	60		80		100		120		ns
$t_{su(D)}$	Data setup time	50		60		80		100		ns
$t_{h(A)}$	Address hold time	0		0		0		0		ns
$t_{h(D)}$	Data hold time	5		10		10		10		ns

switching characteristics over recommended voltage range, T_A = 0°C to 70°C

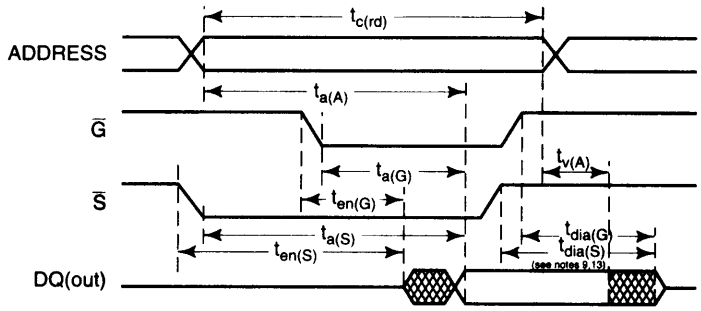
PARAMETER		TMS4016-12		TMS4016-15		TMS4016-20		TMS4016-25		UNIT
		MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	
$t_{a(A)}$	Access time from address		120		150		200		250	ns
$t_{a(S)}$	Access time from chip select low		60		75		100		120	ns
$t_{a(G)}$	Access time from output enable low		50		60		80		100	ns
$t_{v(A)}$	Output data valid after address change	10		15		15		15		ns
$t_{dis(S)}$	Output disable time after chip select high		40		50		60		80	ns
$t_{dis(G)}$	Output disable time after output enable high		40		50		60		80	ns
$t_{dis(W)}$	Output disable time after write enable low		50		60		60		80	ns
$t_{en(S)}$	Output enable time after chip select low	5		5		10		10		ns
$t_{en(G)}$	Output enable time after output enable low	5		5		10		10		ns
$t_{en(W)}$	Output enable time after write enable high	5		5		10		10		ns

NOTES: 3. C_L = 100pF for all measurements except $t_{dis(W)}$ and $t_{en(W)}$.
C_L = 5 pF for $t_{dis(W)}$ and $t_{en(W)}$.
4. t_{dis} and t_{en} parameters are sampled and not 100% tested.

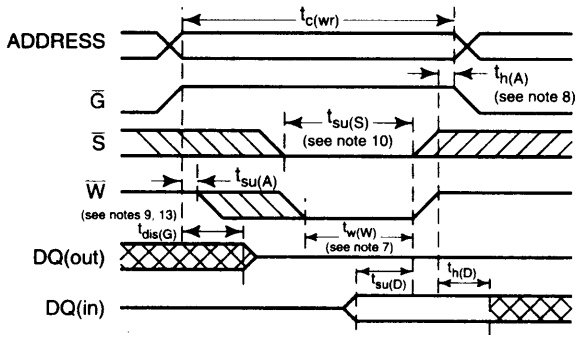
FIGURE 9–5 (a) The AC characteristics of the TMS4016 SRAM. (b) The timing diagrams of the TMS4016 SRAM. (Courtesy of Texas Instruments Incorporated.)

As with the SRAM, the R/W pin writes data to the DRAM when a logic 0, but there is no pin labeled G or enable. There also is no S (select) input to the DRAM. As mentioned, the $\overline{CAS}$ input selects the DRAM. If selected, the DRAM is written if R/W = 0 and read if R/W = 1.

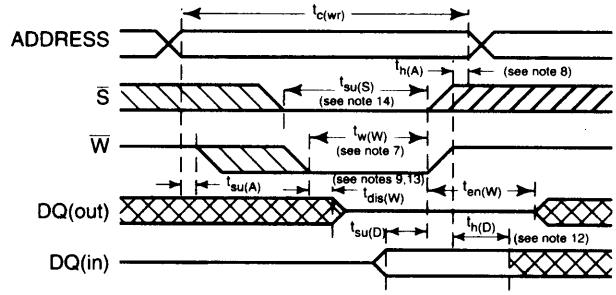
timing waveform of read cycle (see note 5)



timing waveform of write cycle no. 1 (see note 6)

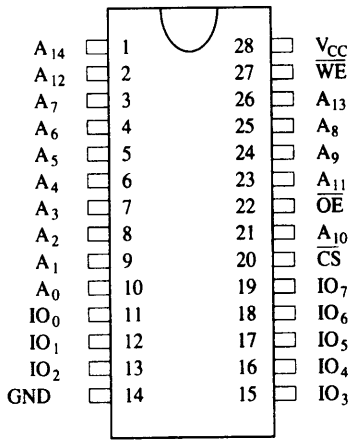


timing waveform of write cycle no. 2 (see notes 6 and 11)



NOTES: 5. $\overline{W}$ is high Read Cycle.
6. $\overline{W}$ must be high during all address transitions.
7. A write occurs during the overlap of a low $\overline{S}$ and a low $\overline{W}$.
8. $t_{h(A)}$ is measured from the earlier of $\overline{S}$ or $\overline{W}$ going high to the end of the write cycle.
9. During this period, I/O pins are in the output state so that the input signals of opposite phase to the outputs must not be applied.
10. If the Slow transition occurs simultaneously with the $\overline{W}$ low transitions or after the $\overline{W}$ transition, output remains in a high impedance state.
11. G is continuously low ($G = V_{IL}$).
12. If $\overline{S}$ is low during this period, I/O pins are in the output state. Data input signals of opposite phase to the outputs must not be applied.
13. Transition is measured ± 200 mV from steady-state voltage.
14. If the $\overline{S}$ low transition occurs before the W low transition, then the data input signals of opposite phase to the outputs must not be applied for the duration of $t_{dis(W)}$ after the $\overline{W}$ low transition.

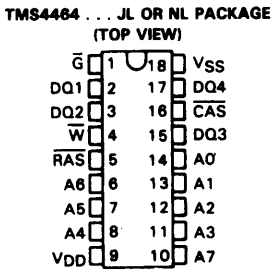
FIGURE 9–5 *(continued)*



PIN FUNCTION

A_0 - A_{14}	Addresses
IO_0 - IO_7	Data connections
$\overline{CS}$	Chip select
$\overline{OE}$	Output enable
$\overline{WE}$	Write enable
V_{CC}	+5V Supply
GND	Ground

FIGURE 9–6 Pin diagram of the 62256, 32K x 8 static RAM.



(a)

PIN NOMENCLATURE	
A0-A7	Address Inputs
$\overline{CAS}$	Column Address Strobe
DQ1-DQ4	Data-In/Data-Out
$\overline{G}$	Output Enable
$\overline{RAS}$	Row Address Strobe
V_{DD}	+5-V Supply
V_{SS}	Ground
$\overline{W}$	Write Enable

(b)

FIGURE 9–7 The pin-out of the TMS4464, 64K x 4 dynamic RAM (DRAM). (Courtesy of Texas Instruments Incorporated.)

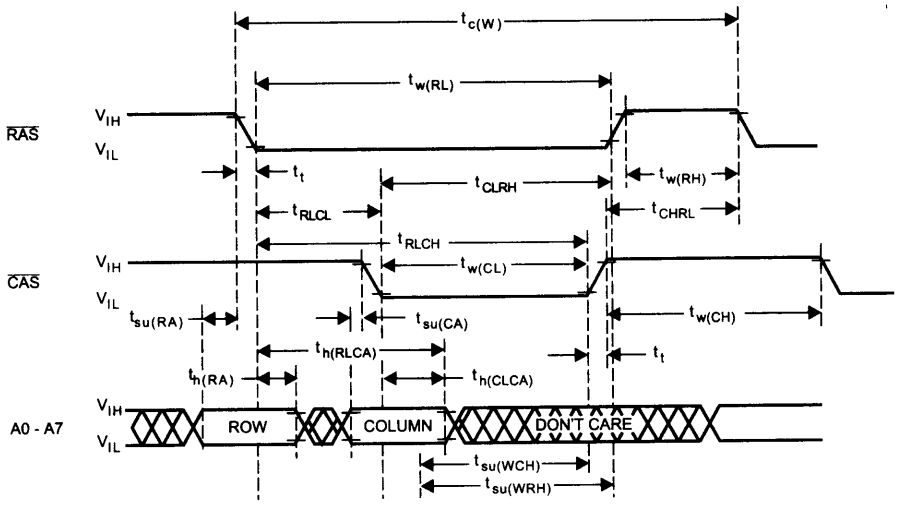


FIGURE 9–8 $\overline{RAS}$, $\overline{CAS}$, and address input timing for the TMS4464 DRAM. (Courtesy of Texas Instruments Incorporated.)

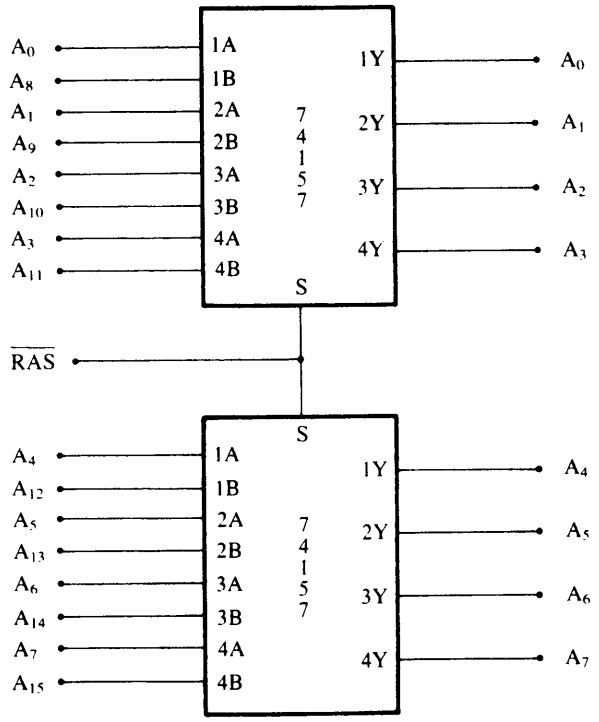


FIGURE 9–9 Address multiplexer for the TMS4464 DRAM.

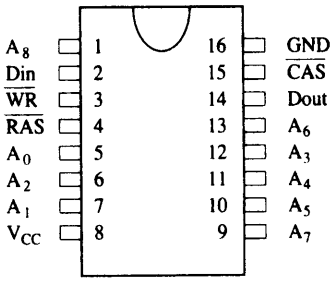
Figure 9–10 shows the pin-out of the 41256 dynamic RAM. This device is organized as a 256K × 1 memory, requiring as little as 70 ns to access data.

9–2 ADDRESS DECODING

In order to attach a memory device to the microprocessor, it is necessary to decode the address sent from the microprocessor. Decoding makes the memory function at a unique section or partition of the memory map. Without an address decoder, only one memory device can be connected to a microprocessor, which would make it virtually useless. In this section, we describe a few of the more common address-decoding techniques, as well as the decoders that are found in many systems.

Why Decode Memory?

When the 8088 microprocessor is compared to the 2716 EPROM, a difference in the number of address connections is apparent—the EPROM has 11 address connections and the microprocessor has 20. This means that the microprocessor sends out a 20-bit memory address whenever it reads or writes data. Because the EPROM has only 11 address pins, there is a mismatch that must be corrected. If only 11 of the 8088's address pins are connected to the



PIN FUNCTIONS

A_0 - A_8	Addresses
Din	Data in
Dout	Data out
$\overline{CAS}$	Column Address Strobe
$\overline{RAS}$	Row Address Strobe
$\overline{WR}$	Write enable
V_{CC}	+5V Supply
GND	Ground

FIGURE 9–10 The 41256 dynamic RAM organized as a 256K x 1 memory device.

memory, the 8088 will see only 2K bytes of memory instead of the 1M bytes that it "expects" the memory to contain. The decoder corrects the mismatch by decoding the address pins that do not connect to the memory component

Simple NAND Gate Decoder

When the 2K x 8 EPROM is used, address connections A10–A0 of the 8088 are connected to address inputs A10–A0 of the EPROM. The remaining nine address pins (A19–A11) are connected to the inputs of a NAND gate decoder (see Figure 9–11). The decoder selects the EPROM from one of the many 2K-byte sections of the entire 1M-byte address range of the 8088 microprocessor.

In this circuit, a single NAND gate decodes the memory address. The output of the NAND gate is a logic 0 whenever the 8088 address pins attached to its inputs (A19–A11) are all logic 1s. The active low, logic 0 output of the NAND gate decoder is connected to the CE input pin that selects **(enables)** the EPROM. Recall that whenever CE is a logic 0, data will be read from the EPROM only if OE is also a logic 0. The OE pin is activated by the 8088 RD signal or the MRDC **(memory read control)** signal of other family members.

If the 20-bit binary address, decoded by the NAND gate, is written so that the leftmost nine bits are 1s and the rightmost 11 bits are don't cares (X), the actual address range of the EPROM can be determined. (A *don't care* is a logic 1 or a logic 0, whichever is appropriate.)

Example 9–1 illustrates how the address range for this EPROM is determined by writing down the externally decoded address bits (A19–A11) and the address bits decoded by the EPROM (A10–A0) as don't cares. As the example illustrates, the don't cares are first written as 0s to locate the lowest address and then as 1s to find the highest address. Example 9–1 also shows these binary boundaries as hexadecimal addresses. Here, the 2K EPROM is decoded at memory address locations FF800H–FFFFFH. Notice that this is a 2K-byte section of the memory and is also located at the reset location for the 8086/8088 (FFFF0H), the most likely place for an EPROM.

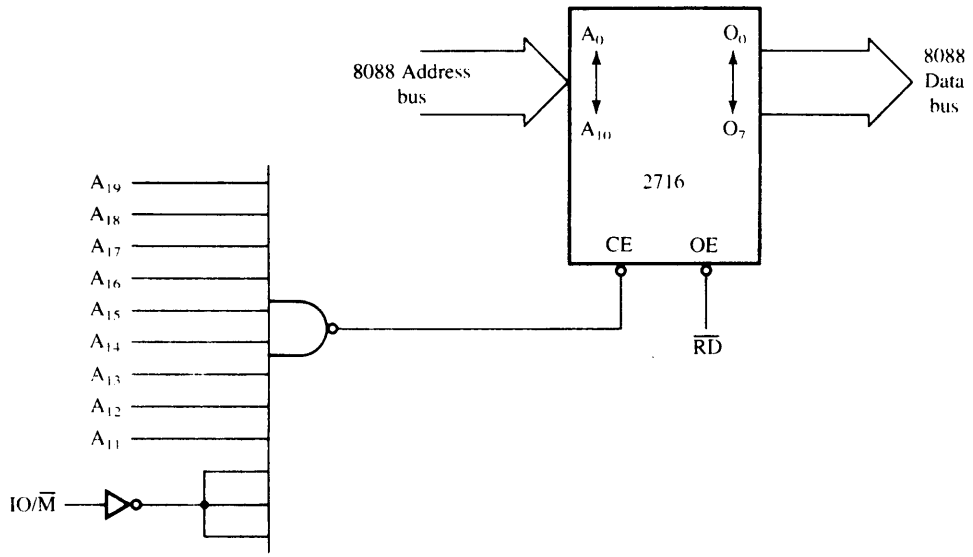


FIGURE 9–11 A simple NAND gate decoder used to select a 2716 EPROM memory component for memory locations FF800H–FFFFFH.

EXAMPLE 9–1

```
1111 1111 1XXX XXXX XXXX

          or

1111 1111 1000 0000 0000 = FF800H
          to
1111 1111 1111 1111 1111 = FFFFFH
```

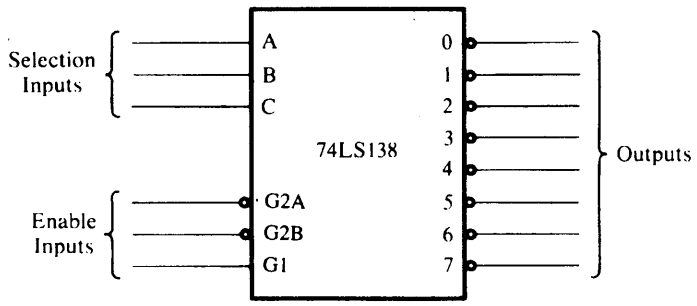
Although this example serves to illustrate decoding, NAND gates are rarely used to decode memory because each memory device requires its own NAND gate decoder. Because of the excessive cost of the NAND gate decoder and inverters that are often required, this option requires that an alternate be found.

The 3-to-8 Line Decoder (74LS138)

One of the more common, although not only, integrated circuit decoders found in many microprocessor-based systems is the 74LS138 3-to-8 line decoder. Figure 9–12 illustrates this decoder and its truth table.

The truth table shows that only one of the eight outputs ever goes low at any time. For any of the decoder's outputs to go low, the three enable inputs (G2A, G2B, and G1) must all be active. To be active, the G2A and G2B inputs must both be low (logic 0), and G1 must be high (logic 1). Once the 74LS138 is enabled, the address inputs (C, B, and A) select which output pin goes low. Imagine eight EPROM CE inputs connected to the eight outputs of the decoder! This is a very powerful device because it selects eight different memory devices at the same time.

Sample Decoder Circuit. Notice that the outputs of the decoder, illustrated in Figure 9–13, are connected to eight different 2764 EPROM memory devices. Here, the decoder selects eight 8K-byte blocks of memory for a total memory capacity of 64K bytes. This figure also illustrates the address range of each memory device and the common connections to the memory devices. Notice that all of the address connections from the 8088 are connected to this circuit. Also, notice that the decoder's outputs are connected to the CE inputs of the EPROMs, and the RD signal from



Inputs						Outputs							
Enable			Select										
$\overline{G2A}$	$\overline{G2B}$	G1	C	B	A	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$	$\overline{4}$	$\overline{5}$	$\overline{6}$	$\overline{7}$
1	X	X	X	X	X	1	1	1	1	1	1	1	1
X	1	X	X	X	X	1	1	1	1	1	1	1	1
X	X	0	X	X	X	1	1	1	1	1	1	1	1
0	0	1	0	0	0	0	1	1	1	1	1	1	1
0	0	1	0	0	1	1	0	1	1	1	1	1	1
0	0	1	0	1	0	1	1	0	1	1	1	1	1
0	0	1	0	1	1	1	1	1	0	1	1	1	1
0	0	1	1	0	0	1	1	1	1	0	1	1	1
0	0	1	1	0	1	1	1	1	1	1	0	1	1
0	0	1	1	1	0	1	1	1	1	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	1	0

FIGURE 9–12 The 74LS138, 3-to-8 line decoder and function table.

the 8088 is connected to the OE inputs of the EPROMs. This allows only the selected EPROM to be enabled and to send its data to the microprocessor through the data bus whenever RD becomes a logic 0.

In this circuit, a 3-input NAND gate is connected to address bits A19–A17. When all three address inputs are high, the output of this NAND gate goes low and enables input G2B of the 74LS138. Input G1 is connected directly to A16. In other words, in order to enable this decoder, the first four address connections (A19–A16) must all be high.

The address inputs C, B, and A connect to microprocessor address pins A15–A13. These three address inputs determine which output pin goes low and which EPROM is selected whenever the 8088 outputs a memory address within this range to the memory system.

Example 9–2 shows how the address range of the entire decoder is determined. Notice that the range is location F0000H–FFFFFH. This is a 64K-byte span of the memory.

EXAMPLE 9–2

```
1111 XXXX XXXX XXXX XXXX

          or

1111 0000 0000 0000 0000 = F0000H
          to
1111 1111 1111 1111 1111 = FFFFFH
```

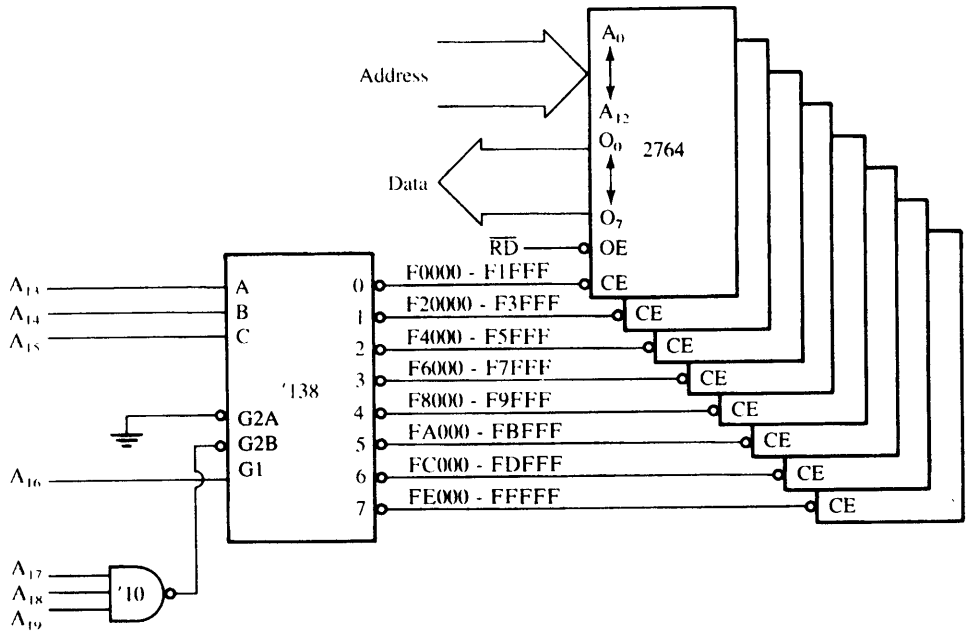


FIGURE 9–13 A circuit that uses eight 2764 EPROMs for a 64K x 8 section of memory in an 8088 microprocessor-based system. The addresses selected in this circuit are F0000H–FFFFFH.

How is it possible to determine the address range of each memory device attached to the decoder's outputs? Again, the binary bit pattern is written down; this time the C, B, and A address inputs are not don't cares. Example 9–3 shows how output 0 of the decoder is made to go low to select the EPROM attached to that pin. Here, C, B, and A are shown as logic 0s.

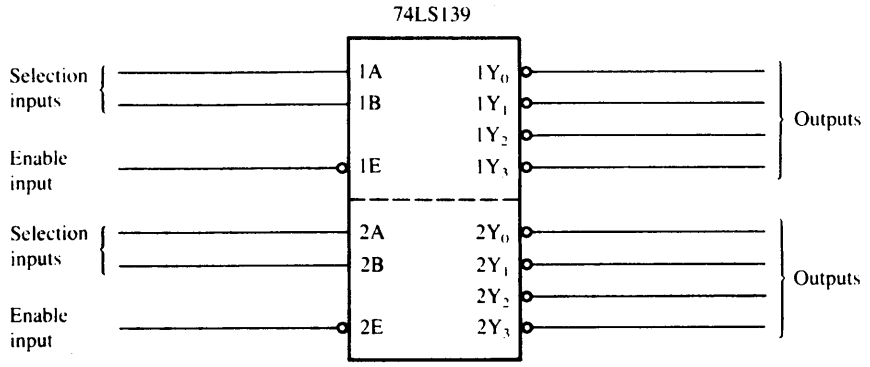
EXAMPLE 9–3

```
     CBA
1111 000X XXXX XXXX XXXX

           or

1111 0000 0000 0000 0000 = F0000H
           to
1111 0001 1111 1111 1111 = F1FFFH
```

If the address range of the EPROM connected to output 1 of the decoder is required, it is determined in exactly the same way as that of output 0. The only difference is that now the C, B, and A inputs contain a 001 instead of a 000 (see Example 9–4). The remaining output address ranges are determined in the same manner by substituting the binary address of the output pin into C, B, and A.

EXAMPLE 9–4

```
     CBA
1111 001X XXXX XXXX XXXX

           or

1111 0010 0000 0000 0000 = F2000H
           to
1111 0011 1111 1111 1111 = F3FFFH
```



Inputs			Outputs			
$\overline{E}$	A	B	$\overline{Y_0}$	$\overline{Y_1}$	$\overline{Y_2}$	$\overline{Y_3}$
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	X	X	1	1	1	1

FIGURE 9-14 The pin-out and truth table of the 74LS139, dual 2-to-4 line decoder.

The Dual 2-to-4 Line Decoder (74LS139)

Another decoder that finds some application is the 74LS139 dual 2-to-4 line decoder. Figure 9–14 illustrates both the pin-out and the truth table for this decoder. The 74LS139 contains two separate 2-to-4 line decoders—each with its own address, enable, and output connections.

PROM Address Decoder

Another, once common, address decoder is the bipolar PROM, used because of its larger number of input connections, which reduces the number of other circuits required in a system memory address decoder. The 74LS138 decoder has six inputs used for address connections. The PROM decoder may have many more inputs for address decoding.

For example, the 82S147 (512 × 8) PROM used as an address decoder has 10 input connections and eight output connections. It can replace the circuit in Figure 9–13 without the extra 3-input NAND gate. This saves space on the printed circuit board and reduces the cost of a system.

Figure 9–15 illustrates this address decoder with the PROM in place. The PROM is a memory device that must be programmed with the correct binary bit pattern to select the eight EPROM memory devices. The PROM itself has nine address inputs that select one of the 512 internal 8-bit memory locations. The remaining input (CE) must be grounded because if this PROM's outputs float to their high-impedance state, one or more of the EPROMs might be selected by noise impulses in the system.

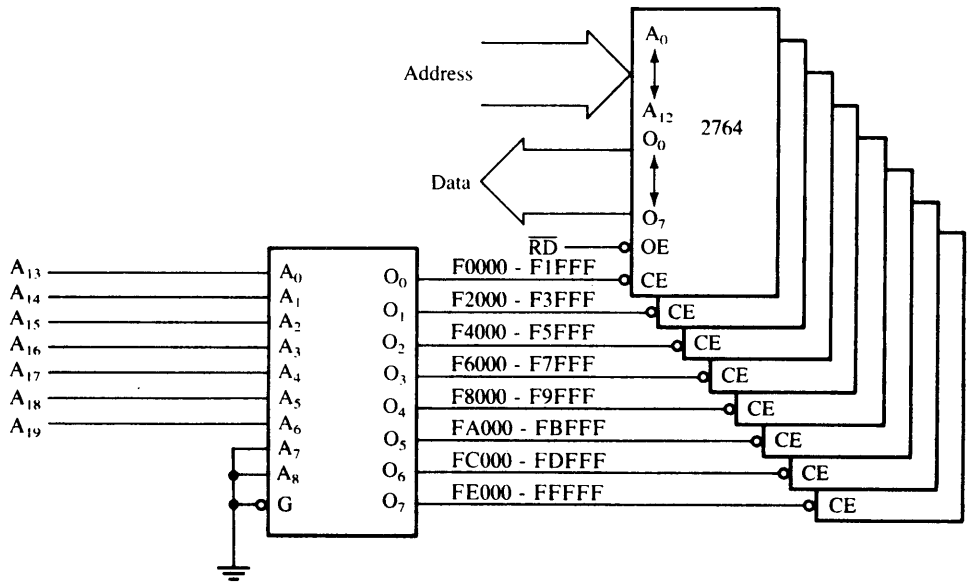


FIGURE 9–15 A memory system using the TPB28L42, 512 x 8 PROM as an address decoder.

TABLE 9–1 The 82S147 PROM programming pattern for the circuit of Figure 9–15.

Inputs										*Outputs*							
OE	A8	A7	A6	A5	A4	A3	A2	A1	A0	O0	O1	O2	O3	O4	O5	O6	O7
0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1
0	0	0	1	1	1	1	0	0	1	1	0	1	1	1	1	1	1
0	0	0	1	1	1	1	0	1	0	1	1	0	1	1	1	1	1
0	0	0	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1
0	0	0	1	1	1	1	1	0	0	1	1	1	1	0	1	1	1
0	0	0	1	1	1	1	1	0	1	1	1	1	1	1	0	1	1
0	0	0	1	1	1	1	1	1	0	1	1	1	1	1	1	0	1
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
all	other combinations									1	1	1	1	1	1	1	1

Table 9–1 illustrates the binary bit pattern programmed into each PROM location in order to select the eight different EPROMs. The main advantage of using a PROM is that the address map is easily changed in the field. Because the PROM comes with all the locations programmed as logic 1s, only eight of the 512 locations must be programmed. This saves valuable time for the manufacturer.

9–3 8088 AND 80188 (8-BIT) MEMORY INTERFACE

This text contains separate sections on memory interfacing for the 8088 and 80188 with their 8-bit data buses; the 8086, 80186, 80286, and 80386SX with their 16-bit data buses; the 80386DX and 80486 with their 32-bit data buses; the Pentium–Pentium 4 with their 64-bit data buses. Separate sections are provided because the methods used

to address the memory are slightly different in microprocessors that contain different data bus widths. Hardware engineers or technicians who wish to broaden their expertise in interfacing 16-bit, 32-bit, and 64-bit memory interface should cover all sections. This section is much more complete than the section on the 16- and 32-bit memory interface, which covers only material not covered in the 8088/80188 section. In this section, we examine the memory interface to both RAM and ROM.

Basic 8088/80188 Memory Interface

The 8088 and 80188 microprocessors have an 8-bit data bus, which makes them ideal to connect to the common 8-bit memory devices available today. The 8-bit memory size makes the 8088, and especially the 80188, ideal as a simple controller. For the 8088/80188 to function correctly with the memory, however, the memory system must decode the address to select a memory component. It must also use the RD, WR, and IO/M control signals provided by the 8088/80188 to control the memory system.

The minimum mode configuration is used in this section and is essentially the same as the maximum mode system for memory interface. The main difference is that, in maximum mode, the IO/M signal is combined with RD to generate the MRDC signal, and IO/M is combined with WR to generate the MWTC signal. The maximum mode control signals are developed inside the 8288 bus controller. In minimum mode, the memory sees the 8088 or the 80188 as a device with 20 address connections (A19–A0), eight data bus connections (AD7–AD0), and the control signals IO/M, RD, and WR.

Interfacing EPROM to the 8088. You will find this section very similar to Section 9–2 on decoders. The only difference is that, in this section, we discuss wait states and the use of the IO/M signal to enable the decoder.

Figure 9–16 illustrates an 8088/80188 microprocessor connected to eight 2732 EPROMs, 4K × 8 memory devices. The 2732 has one more address input (A11) than the 2716, and twice the memory. The device in this

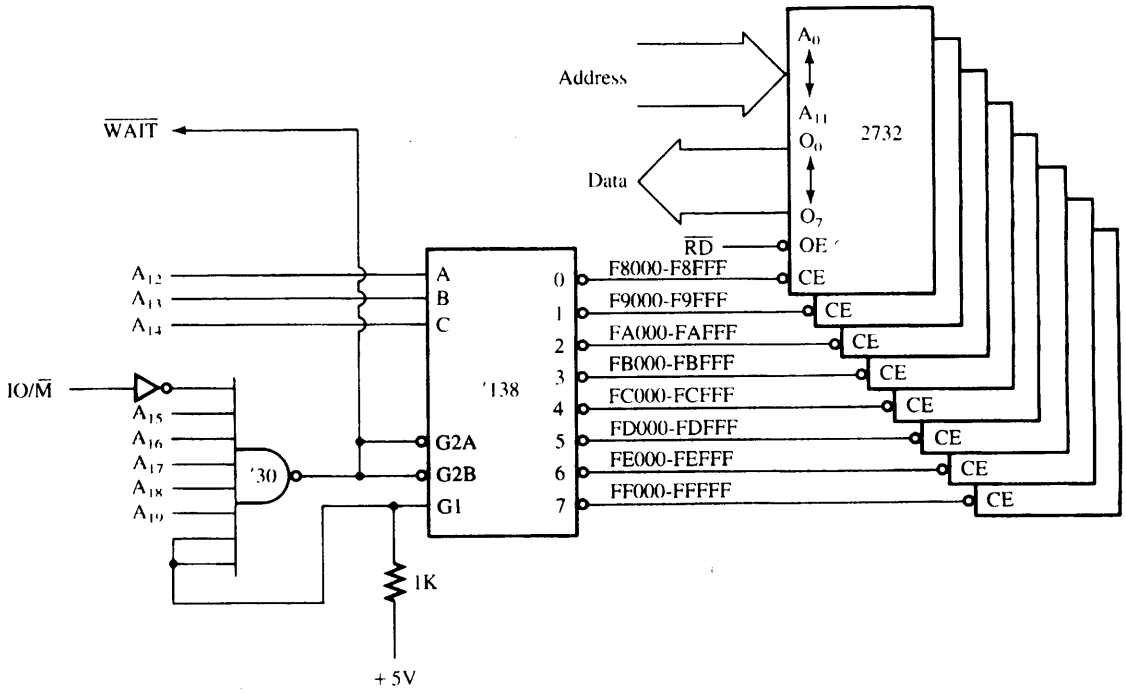


FIGURE 9–16 Eight 2732 EPROMs interfaced to the 8088 microprocessor. Note that the output of the NAND gate is used to cause a wait state whenever this section of the memory is selected.

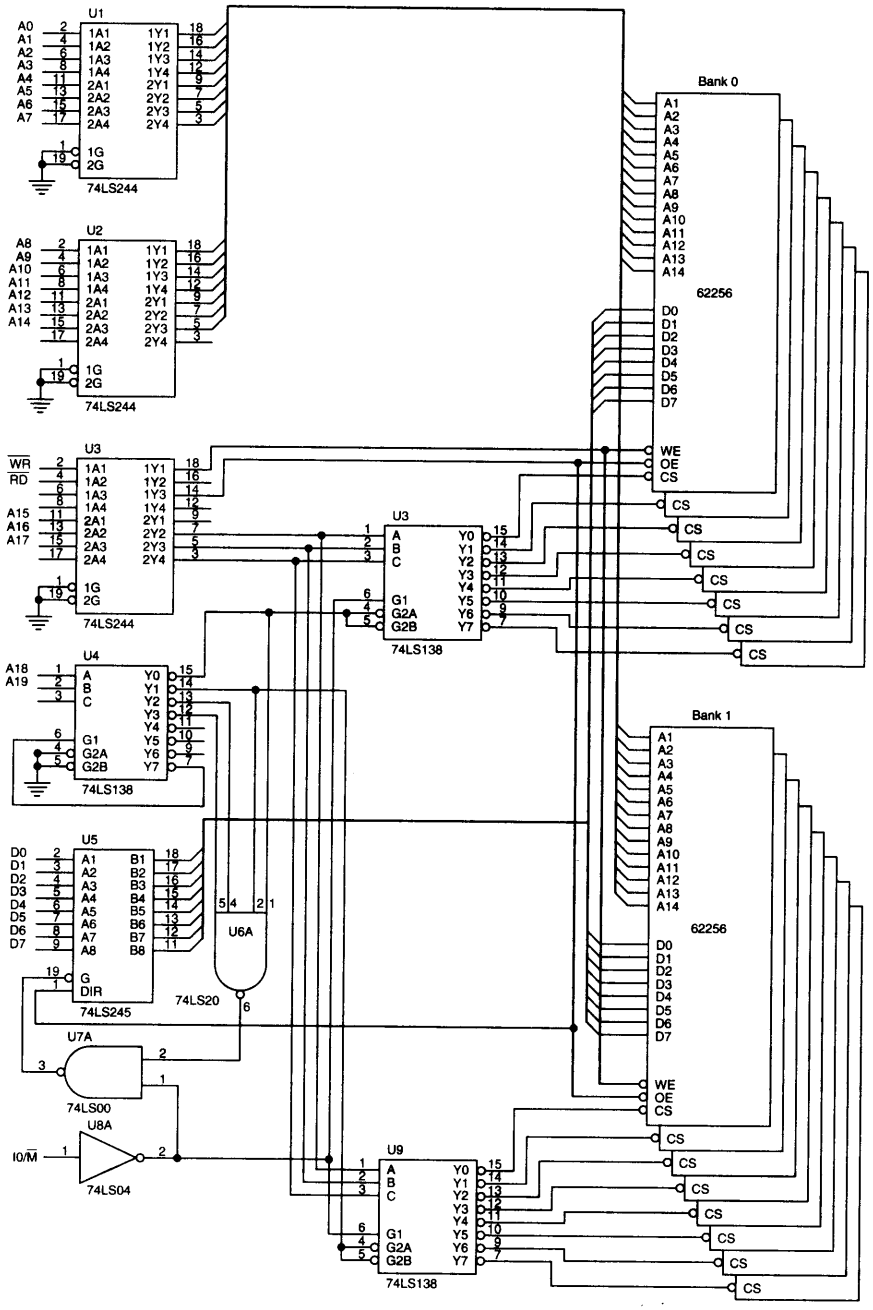


FIGURE 9–17 A 512K byte static memory system using 16 62255 SRAMs.

illustration decodes eight 4K × 8 blocks of memory, for a total of 32K × 8 bits of the physical address space for the 8088/80188.

The decoder (74LS138) is connected a little differently than might be expected because the slower version of this type of EPROM has a memory access time of 450 ns. Recall from Chapter 8 that when the 8088 is operated with a 5 MHz clock, it allows 460 ns for the memory to access data. Because of the decoder's added time delay (12 ns), it is impossible for this memory to function within 460 ns. In order to correct this problem, we must add a NAND gate to generate a signal to enable the decoder and a signal for the wait state generator, covered in Chapter 8. (Note that the 80188 can internally insert from 0–15 wait states without any additional external hardware, so it does not require this NAND gate.) With a wait state inserted every time this section of the memory is accessed, the 8088 will allow 660 ns for the EPROM to access data. Recall that an extra wait state adds 200 ns (1 clock) to the access time. The 660 ns is ample time for a 450 ns memory component to access data, even with the delays introduced by the decoder and any buffers added to the data bus.

Notice that the decoder is selected for a memory address range that begins at location F8000H and continues through location FFFFFH—the upper 32K bytes of memory. This section of memory is an EPROM because FFFF0H is where the 8088 starts to execute instructions after a hardware reset. We often call location FFFF0H the **cold-start** location, also called reset vector. The software stored in this section of memory would contain a JMP instruction at location FFFF0H that jumps to location F8000H so the remainder of the program can execute.

Interfacing RAM to the 8088. RAM is a little easier to interface than EPROM because most RAM memory components do not require wait states. An ideal section of the memory for the RAM is the very bottom, which contains vectors for interrupts. Interrupt vectors (discussed in more detail in Chapter 11) are often modified by software packages, so it is rather important to encode this section of the memory with RAM.

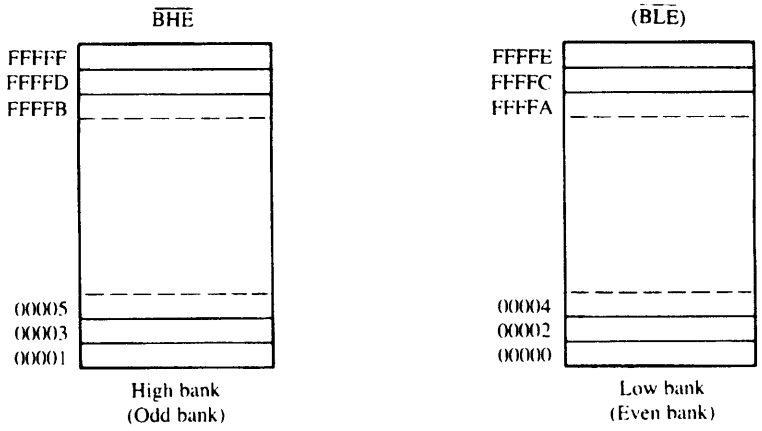
In Figure 9–17, 16 62256 32K × 8 static RAMs are interfaced to the 8088, beginning at memory location 00000H. This circuit board uses two decoders to select the 16 different RAM memory components and a third to select the other decoders for the appropriate memory sections. Sixteen 32K RAMs fill memory from location 00000H through location 7FFFFH, for 512K bytes of memory.

The first decoder (U4) in this circuit selects the other two decoders. An address beginning with 00 selects decoder U3 and an address that begins with 01 selects decoder U9. Notice that extra pins remain at the output of decoder U4 for future expansion. These pins allow more 256K × 8 blocks of RAM, for a total of 1M × 8, simply by adding the RAM and the additional secondary decoders.

Also notice from the circuit in Figure 9–17 that all the address inputs to this section of memory are buffered, as are the data bus connections and control signals RD and WR. Buffering is important when many devices appear on a single board or in a single system. Suppose that three other boards like this are plugged into a system. Without the buffers on each board, the load on the system address, data, and control buses would be enough to prevent proper operation. (Excessive loading causes the logic 0 output to rise above the 0.8 V maximum allowed in a system.) Buffers are normally used if the memory will contain additions at some future date. If the memory will never grow, then buffers may not be needed.

9–4 8086 (16-BIT) MEMORY INTERFACE

The 8086, microprocessors differ from the 8088 in three ways: (1) the data bus is 16 bits wide instead of 8 bits wide as on the 8088, (2) the IO/M pin of the 8088 is replaced with an M/IO pin, and (3) there is a new control signal called bus high enable (BHE). The address bit A0 or BLE is also used differently. (Because this section is based on information provided in Section 9–3, it is extremely important that you read the previous section first.) A few other differences exist between the 8086 and the 80286/80386SX. The 80286/80386SX contains a 24-bit address bus (A23–A0) instead of the 20-bit address bus (A19–A0) of the 8086. The 8086 contain an M/IO signal while the 80286 system and 80386SX microprocessor contain control signals MRDC and MWTC instead of RD and WR. The 8086 has a multiplexed address and Data bus the 80286, 80386 data and address bus are not multiplexed.



Note: A_0 is labeled $\overline{BLE}$ (Bus low enable) on the 80386SX.

FIGURE 9–18 The high (odd) and low (even) 8-bit memory banks of the 8086 microprocessors.

16-Bit Bus Control

The data bus of the 8086 is twice as wide as the bus for the 8088. This wider data bus presents us with a unique set of problems that have not been encountered before. The 8086 must be able to write data to any 16-bit location—or any 8-bit location. This means that the 16-bit data bus must be divided into two separate sections **(banks)** that are eight bits wide so that the microprocessor can write to either half (8-bit) or both halves (16-bit). Figure 9–18 illustrates the two banks of the memory. One bank **(low bank)** holds all the even-numbered memory locations, and the other bank **(high bank)** holds all the odd-numbered memory locations.

The 8086 uses the BHE signal (high bank) and the A0 address bit or BLE (Bus low enable) to select one or both banks of memory used for the data transfer. Table 9–2 depicts the logic levels on these two pins and the bank or banks selected.

Bank selection is accomplished in two ways: (1) a separate write signal is developed to select a write to each bank of the memory, or (2) separate decoders are used for each bank. As a careful comparison reveals, the first technique is by far the least costly approach to memory interface for the 8086 microprocessors.

Separate Bank Decoders. The use of separate bank decoders is often the least effective way to decode memory addresses for the 8086, 80286, and 80386SX microprocessors. This method is sometimes used, but it is difficult to understand why in most cases. One reason may be to conserve energy, because only the bank or banks selected are enabled. This is not always the case with the separate bank read and write signals that are discussed later.

Figure 9–19 illustrates two 74LS138 decoders used to select 64K RAM memory components. Here, decoder U2 has the BLE pin (A0) attached to G2A, and decoder U3 has the BHE signal attached to its G2A input. Because

TABLE 9–2 Memory bank selection using BHE and BLE (A0).

BHE	*BLE (A0)*	*Function*
0	0	Both banks enabled for a 16-bit transfer
0	1	High bank enabled for an 8-bit transfer
1	0	Low bank enabled for an 8-bit transfer
1	1	No banks enabled

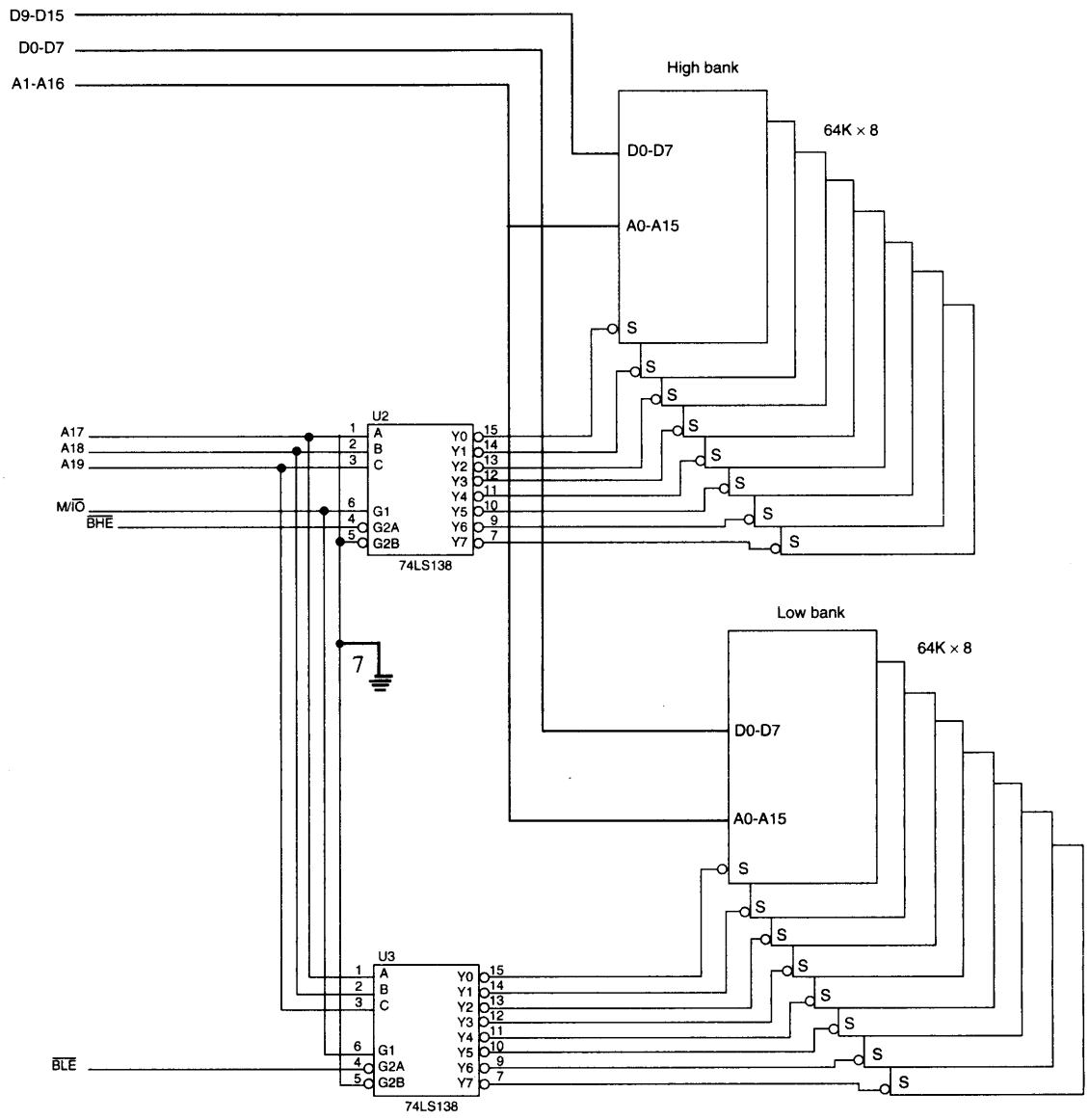


FIGURE 9–19 Separate bank decoders.

the decoder will not activate until all of its enable inputs are active, decoder U2 activates only for a 16-bit operation or an 8-bit operation from the low bank. Decoder U3 activates for a 16-bit operation or an 8-bit operation to the high bank. These two decoders and the 16 (64K-byte) RAMs they control represent a 1M range of the memory system.

Notice in Figure 9–19 that the A0 address pin does not connect to the memory as it is connected to U3 as BLE. Also notice that address bus bit position A1 is connected to memory address input A0, A2 is connected to A1, and so forth. The reason is that A0 from the 8086 (or BLE from the 80286/80386SX) is already connected to decoder U3 and does not need to be connected again to the memory. If A0 or BLE is attached to the A0 address pin of memory, every other memory location in each bank of memory would be used. This means that half of the memory is wasted if A0 or BLE is connected to A0.

Separate Bank Write Strobes. The most effective way to handle bank selection is to develop a separate write strobe for each memory bank. This technique requires only one decoder to select a 16-bit wide memory, which often saves money and reduces the number of components in a system.

Why not also generate separate read strobes for each memory bank? This is usually unnecessary because the 8086, 80286, and 80386SX microprocessors read only the byte of data that they need at any given time from half of the data bus. If 16-bit sections of data are always presented to the data bus during a read, the microprocessor ignores the 8-bit section that it doesn't need, without any conflicts or special problems.

Figure 9–20 depicts the generation of separate 8086 write strobes for the memory. Here, a 74LS32 OR gate combines A0 with WR for the low bank selection signal (LWR), and BHE combines with WR for the high bank selection signal (HWR). Write strobes for the 80286/80386SX are generated by using the MWTC signal instead of WR.

A memory system that uses separate write strobes is constructed differently from either the 8-bit system (8088) or the system using separate memory banks. Memory in a system that uses separate write strobes is decoded as 16-bit wide memory. For example, suppose that a memory system will contain 64K bytes of SRAM memory. This memory requires two 32K byte memory devices (62256) so that a 16-bit wide memory can be constructed. Because the memory is 16 bits wide and another circuit generates the bank write signals, address bit A0 becomes a don't care. In fact, A0 is not even a pin on the 80386SX microprocessor.

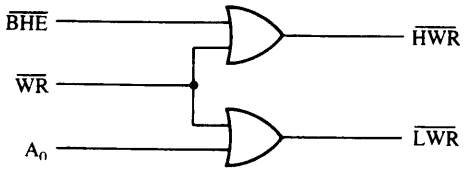


FIGURE 9–20 The memory bank write selection input signals: HWR (high bank write) and LWR (low bank write).

Example 9–5 shows how a 16-bit wide memory stored at locations 060000H–06FFFFH is decoded for the 80286 or 80386 microprocessor. Memory in this example is decoded, so bit A0 is a don't care for the decoder. Bit

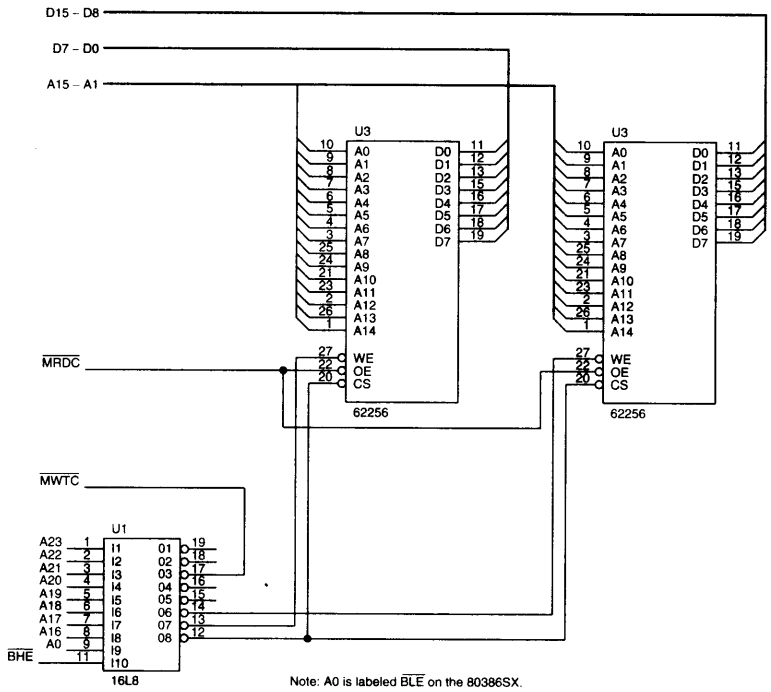


FIGURE 9–21 A 16-bit memory decoder that places memory at locations 060000H–06FFFFH.

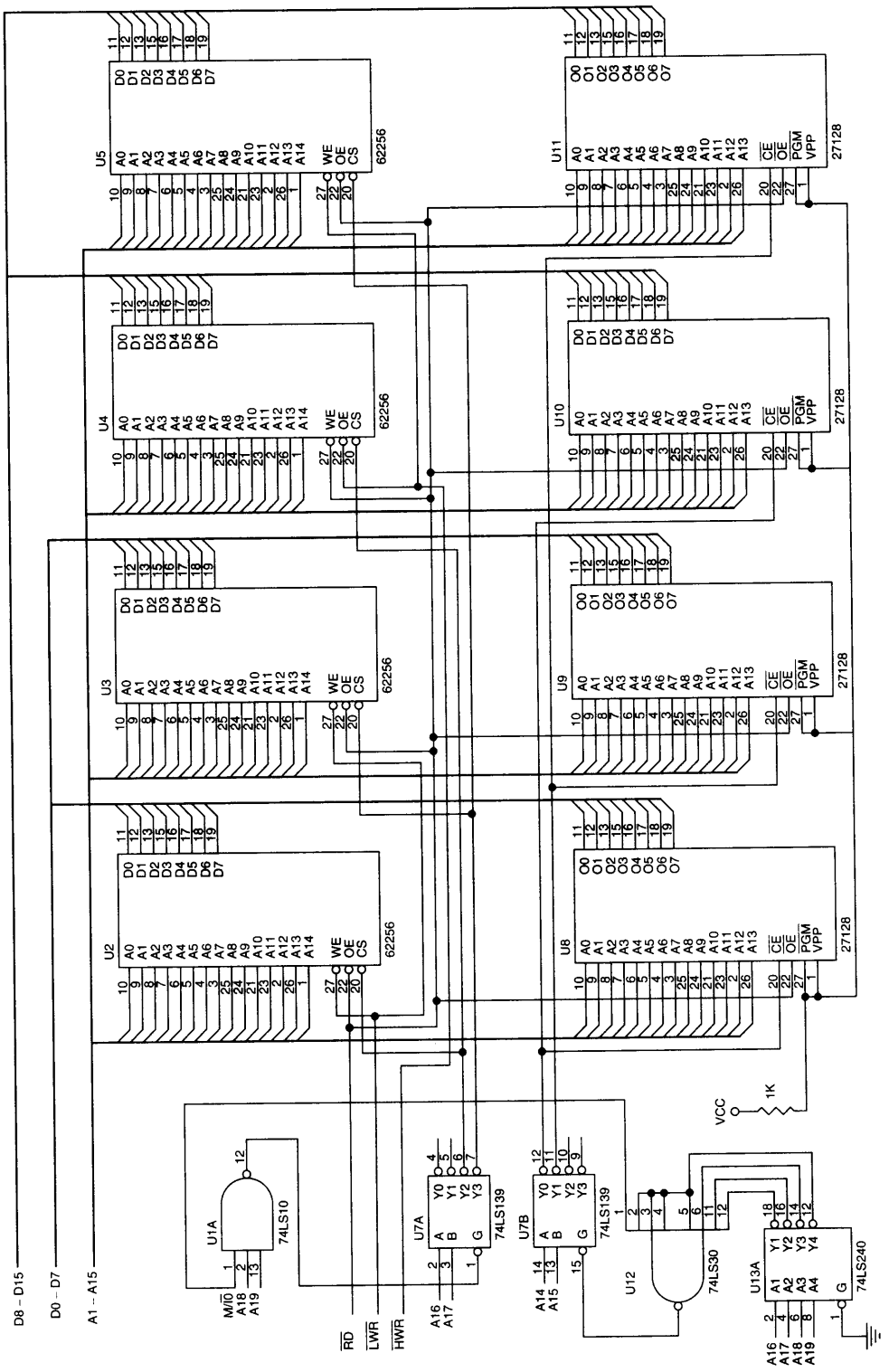


FIGURE 9–22 A memory system for the 8086 that contains a 64K-byte EPROM and a 128K-byte SRAM.

positions A1–A15 are connected to memory component address pins A0–A14. The decoder (PAL16L8) enables both memory devices by using address connection A23–A15 to select memory whenever address 06XXXXH appears on the address bus.

EXAMPLE 9–5

```
0000 0110 0000 0000 0000 0000 = 060000H
            to
0000 0110 1111 1111 1111 1111 = 06FFFFH

0000 0110 XXXX XXXX XXXX XXXX = 06XXXXH
```

Figure 9–21 illustrates this simple circuit by using a PAL16L8 to both decode memory and generate the separate write strobe. The program for the PAL16L8 decoder is illustrated in Example 9–6. Notice that not only is the memory selected, but both the lower and upper write strobes are also generated by the PAL.

EXAMPLE 9–6

```
; pins 1    2    3    4    5    6    7    8    8  10
       A23  A22  A21  A20  A19  A18  A17  A16  A0 GND
;pins 11  12  13  14  15 16 17   18 19 20
      BHE SEL LWR HWR NC NC MWTC NC NC VCC

EQUATIONS

/SEL = /A23 * /A22 * /A21 * /A20 * /A19 * A18 * A17 * /A16
/LWR = /MWTC * /AO
/HWR = /MWTC * /BHE
```

Figure 9–22 depicts a small memory system for the 8086 microprocessor that contains an EPROM section and a RAM section. Here, there are four 27128 EPROMs (16K × 8) that compose a 32K × 16-bit memory at location F0000–FFFFFH and four 62256 (32K × 8) RAMs that compose an 64K × 16-bit memory at location 00000H–1FFFFH. (Remember that even though the memory is 16 bits wide, it is still numbered in bytes.)

This circuit uses a 74LS139 dual 2-to-4 line decoder that selects EPROM with one half and RAM with the other half. It decodes memory that is 16 bits wide; not eight bits, as before. Notice that the RD strobe is connected to all the EPROM OE inputs and all RAM G input pins. This is done because even if the 8086 is reading only eight bits of data, the application of the remaining eight bits to the data bus has no effect on the operation of the 8086.

The LWR and HWR strobes are connected to different banks of the RAM memory. Here, it does matter whether the microprocessor is doing a 16-bit or an 8-bit write. If the 8086 writes a 16-bit number to memory, both LWR and HWR go low and enable the W pins in both memory banks. But if the 8086 does an 8-bit write, only one of the write strobes goes low, writing to only one memory bank. Again, the only time that the banks make a difference is for a memory write operation.

Notice that an EPROM decoder signal is sent to the 8086 wait state generator because EPROM memory usually requires a wait state. The signal comes from the NAND gate used to select the EPROM decoder section, so that if EPROM is selected, a wait state is requested.

9–5 SUMMARY

1. All memory devices have address inputs; data inputs and outputs, or just outputs; a pin for selection; and one or more pins that control the operation of the memory.
2. Address connections on a memory component are used to select one of the memory locations within the device. Ten address pins have 1024 combinations and therefore are able to address 1024 different memory locations.

3. Data connections on a memory are used to enter information to be stored in a memory location and also to retrieve information read from a memory location. Manufacturers list their memory as, for example, 4K × 4, which means that the device has 4K memory locations (4096) and that four bits are stored in each location.
4. Memory selection is accomplished via a chip selection pin (CS) on many RAMs or a chip enable pin (CE) on many EPROM or ROM memories.
5. Memory function is selected by an output enable pin (OE) for reading data, which normally connects to the system read signal (RD or MRDC). The write enable pin (WE), for writing data, normally connects to the system write signal (WR or MWTC).
6. An EPROM memory is programmed by an EPROM programmer and can be erased if exposed to ultraviolet light. Today, EPROMs are available in sizes from 1K × 8 all the way up to 128K × 8 and larger.
7. Static RAM (SRAM) retains data for as long as the system power supply is attached. These memory types are available in sizes up to 128K × 8.
8. Memory address decoders select an EPROM or RAM at a particular area of the memory. Commonly found address decoders include the 74LS138 3-to-8 line decoder, the 74LS139 2-to-4 line decoder, and programmed selection logic in the form of a PROM or PLD.
9. The PROM and PLD address decoders for microprocessors like the 8088 through the Pentium 4 reduce the number of integrated circuits required to complete a functioning memory system.
10. The 8088 minimum mode memory interface contains 20 address lines, eight data lines, and three control lines: RD, WR, and IO/M. The 8088 memory functions correctly only when all these lines are used for memory interface.
11. The access speed of the EPROM must be compatible with the microprocessor to which it is interfaced. Many EPROMs available today have an access time of 450 ns, which is too slow for the 5 MHz 8088. In order to circumvent this problem, a wait state is inserted to increase memory access time to 660 ns.
12. The 8086 memory interface has a 16-bit data bus and contains an M/IO control pin, whereas the 8088 has an 8-bit data bus and contains an IO/M pin. In addition to these changes, there is an extra control signal, bus high enable (BHE).
13. The 8086 memory is organized in two 8-bit banks: high bank and low bank. The high bank of memory is enabled by the BHE control signal and the low bank is enabled by the A0 address signal or by the BLE control signal.
14. Two common schemes for selecting the banks in an 8086 based system include (1) a separate decoder for each bank and (2) separate WR control signals for each bank with a common decoder.

9–6 QUESTIONS AND PROBLEMS

1. What types of connections are common to all memory devices?
2. List the number of words found in each memory device for the following numbers of address connections:
   (a) 8
   (b) 11
   (c) 12
   (d) 13
3. List the number of data items stored in each of the following memory devices and the number of bits in each datum:
   (a) 2K × 4
   (b) 1K ⑨1
   (c) 4K ⑨8
   (d) 16K ⑨1
   (e) 64K ⑨4

4. What is the purpose of the CS or CE pin on a memory component?
5. What is the purpose of the OE pin on a memory device?
6. What is the purpose of the WE pin on a RAM?
7. How many bytes of storage do the following EPROM memory devices contain?
   (a) 2708
   (b) 2716
   (c) 2732
   (d) 2764
   (e) 27128
8. Why won't a 450 ns EPROM work directly with a 5 MHz 8088?
9. What can be stated about the amount of time to erase and write a location in a flash memory device?
10. SRAM is an acronym for what type of device?
11. Why are memory address decoders important?
12. Modify the NAND gate decoder of Figure 9–11 to select the memory for address range DF800H–DFFFFH.
13. Modify the NAND gate decoder in Figure 9–11 to select the memory for address range 40000H–407FFH.
14. When the G1 input is high, and G2A and G2B are both low, what happens to the outputs of the 74LS138 3-to-8 line decoder?
15. Modify the circuit of Figure 9–13 to address memory range 70000H–7FFFFH.
16. Modify the circuit of Figure 9–13 to address memory range 40000H–4FFFFH.
17. Describe the 74LS139 decoder.
18. Why is a PROM address decoder often found in a memory system?
19. Reprogram the PROM in Table 9–1 to decode memory address range 80000H–8FFFFH.
20. Reprogram the PROM in Table 9–1 to decode memory address range 30000H–3FFFFH.
21. Modify the circuit of Figure 9–19 by rewriting the PAL program to address memory at locations 40000H–4FFFFH.
22. Modify the circuit of Figure 9–19 by rewriting the PAL program to address memory at locations B0000H–BFFFFH.
23. The RD and WR minimum mode control signals are replaced by what two control signals in the 8086 maximum mode?
34. Modify the circuit of Figure 9–16 to select memory at location 68000–6FFFFH.
25. Modify the circuit of Figure 9–16 to select eight 2764 8K×8 EPROMs at memory location 10000H–1FFFFH.
26. Add another decoder to the circuit of Figure 9–17 so that an additional eight 62256 SRAMs are added at location C0000H–FFFFFH.
27. What is the purpose of the BHE and A0 pins on the 8086 microprocessor?
28. What is the BLE pin and what other pin has it replaced?
29. What two methods are used to select the memory in the 8086 microprocessor?
30. If BHE is a logic 0, then the _____ memory bank is selected.
31. If A0 is a logic 0, then the _____ memory bank is selected.
32. Why don't separate bank read (RD) strobes need to be developed when interfacing memory to the 8086?
33. Modify the circuit of Figure 9–22 so that the EPROM is located at memory range C0000H–CFFFFH and the RAM is located at memory range 30000H–4FFFFH.

CHAPTER 10

Basic I/O Interface

INTRODUCTION

A microprocessor is great at solving problems, but if it can't communicate with the outside world, it is of little worth. This chapter outlines some of the basic methods of communications, both serial and parallel, between humans or machines and the microprocessor.

In this chapter, we first introduce the basic I/O interface and discuss decoding for I/O devices. Then, we provide detail on parallel and serial interfacing, both of which have a variety of applications. As applications, we connect analog-to-digital and digital-to-analog converters, as well as both DC and stepper motors to the microprocessor.

CHAPTER OBJECTIVES

Upon completion of this chapter, you will be able to:

1. Explain the operation of the basic input and output interfaces.
2. Decode an 8-, 16, and 32-bit I/O device so that they can be used at any I/O port address.
3. Define handshaking and explain how to use it with I/O devices.
4. Interface and program the 82C55 programmable parallel interface.
5. Interface LCD displays, LED displays, keyboards, ADC, DAC, and various other devices to the 82C55.
6. Interface and program the 8279 programmable keyboard/display controller.
7. Interface and program the 16550 serial communications interface adapter.
8. Interface and program the 8254 programmable interval timer.
9. Interface an analog-to-digital converter and a digital-to-analog converter to the microprocessor
10. Interface both DC and stepper motors to the microprocessor.

10–1 INTRODUCTION TO I/O INTERFACE

In this section of the text, we explain the operation of the I/O instructions (IN and OUT). We also explain the concept of isolated (sometimes called direct or I/O mapped I/O) and memory-mapped I/O, the basic input and output interfaces, and handshaking. A working knowledge of these topics will make it easier to understand the connection and operation of the programmable interface components and I/O techniques presented in the remainder of this chapter and text.

TABLE 10–1 Input/output instructions.

Instruction	*Data Width*	*Function*
IN AL, p8	8	A byte is input from port p8 into AL
IN AX, p8	16	A word is input from port p8 into AX
IN AL, DX	8	A byte is input from the port addressed by DX into AL
IN AX, DX	16	A word is input from the port addressed by DX into AX
OUT p8, AL	8	A byte is output from AL to port p8
OUT p8, AX	16	A word is output from AX to port p8
OUT DX, AL	8	A byte is output from AL to the port addressed by DX
OUT DX, AX	16	A word is output from AX to the port addressed by DX

I/O Instructions

The instruction set contains one type of instruction that transfers information to an I/O device (OUT) and another to read information from an I/O device (IN). Table 10–1 lists all versions of each instruction found in the microprocessor's instruction set.

Both the IN and OUT instructions transfer data between an I/O device and the microprocessor's accumulator (AL or AX). The I/O address is stored in register DX as a 16-bit I/O address or in the byte (p8) immediately following the opcode as an 8-bit I/O address. Intel calls the 8-bit form (p8) a **fixed address** because it is stored with the instruction, usually in a ROM. The 16-bit I/O address in DX is called a **variable address** because it is stored in a DX, and then used to address the I/O device.

Whenever data are transferred by using the IN or OUT instruction, the I/O address, often called a **port number** (or simply port), appears on the address bus. The external I/O interface decodes the port number in the same manner that it decodes a memory address. The 8-bit fixed port number (p8) appears on address bus connections A7–A0 with bits A15–A8 equal to 00000000_2. The address connections above A15 are undefined for an I/O instruction. The 16-bit variable port number (DX) appears on address connection A15–A0. This means that the first 256 I/O port addresses (00H–FFH) are accessed by both the fixed and variable I/O instructions, but any I/O address from 0100H–FFFFH is only accessed by the variable I/O address. In many dedicated task systems, only the rightmost eight bits of the address are decoded, thus reducing the amount of circuitry required for decoding. In a PC computer, all 16 address bus bits are decoded with locations 0000H–03XXH, which are the I/O addresses used for I/O inside the PC for the ISA (**industry standard architecture**) bus.

Isolated and Memory-Mapped I/O

There are two different methods of interfacing I/O to the microprocessor: **isolated I/O** and **memory-mapped I/O.** In the isolated I/O scheme, the IN and OUT instructions transfer data between the microprocessor's accumulator or memory and the I/O device. In the memory-mapped I/O scheme, any instruction that references memory can accomplish the transfer. Both isolated and memory-mapped I/O are in use, so both are discussed in this text.

Isolated I/O. The most common I/O transfer technique used in the Intel microprocessor-based system is isolated I/O. The term *isolated* describes how the I/O locations are isolated from the memory system in a separate I/O address space. (Figure 10–1 illustrates both the isolated and memory-mapped address spaces for any Intel 80X86 or Pentium–Pentium 4 microprocessor.) The addresses for isolated I/O devices, called **ports,** are separate from the memory. Because the ports are separate, the user can expand the memory to its full size without using any of memory space for I/O devices. A disadvantage of isolated I/O is that the data transferred between I/O and the microprocessor must be accessed by the IN and OUT instructions. Separate control signals

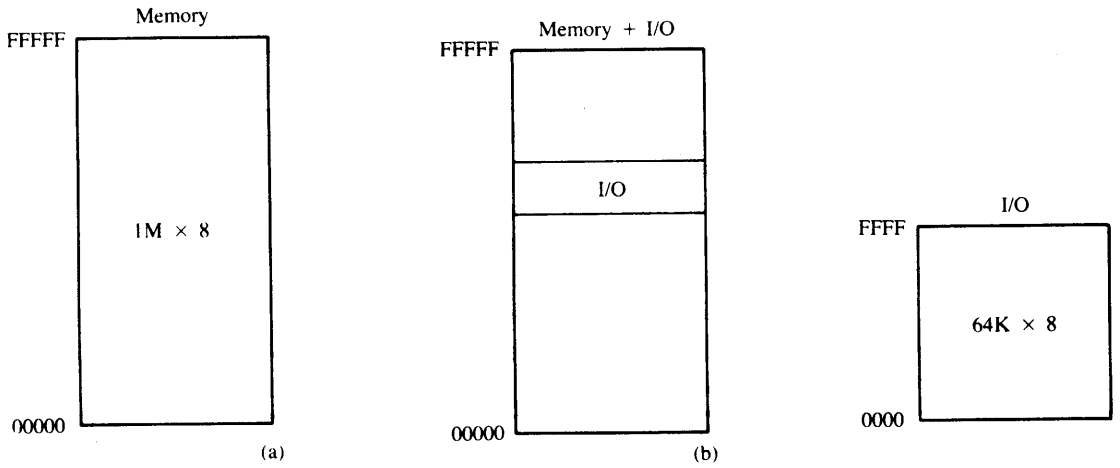


FIGURE 10–1 The memory and I/O maps for the 8086/ 8088 microprocessors. (a) Isolated I/O (b) Memory-mapped I/O.

for the I/O space are developed (using M/IO and W/R), which indicate an I/O read (IORC) or an I/O write (IOWC) operation. These signals indicate that an I/O port address, which appears on the address bus, is used to select the I/O device. In the personal computer, isolated I/O ports are used for controlling peripheral devices. An 8-bit port address is used to access devices located on the system board, such as the timer and keyboard interface, while a 16-bit port is used to access serial and parallel ports as well as video and disk drive systems.

Memory-Mapped I/O. Unlike isolated I/O, memory-mapped I/O does not use the IN and OUT instructions. Instead, it uses any instruction that transfers data between the microprocessor and memory. A memory-mapped I/O device is treated as a memory location in the memory map. The main advantage of memory-mapped I/O is that any memory transfer instruction can be used to access the I/O device. The main disadvantage is that a portion of the memory system is used as the I/O map. This reduces the amount of memory available to applications. Another advantage is that the IORC and IOWC signals have no function in a memory-mapped I/O system and may reduce the amount of circuitry required for decoding.

Personal Computer I/O Map

The personal computer uses part of the I/O map for dedicated functions. Figure 10–2 shows the I/O map for the PC. Note that I/O space between ports 0000H and 03FFH are normally reserved for the computer system and the ISA bus. The I/O ports located at 0400H–FFFFH are generally available for user applications, main-board functions, and the PCI bus. Note that the 80287 arithmetic coprocessor uses I/O address 00F8H–00FFH for communications. For this reason, Intel reserves I/O ports 00F0H–00FFH. The 80386–Pentium 4 use I/O ports 800000F8-800000FFH for communications to their coprocessors. The I/O ports located between 0000H and 00FFH are accessed via the fixed port I/O instructions; the ports located above 00FFH are accessed via the variable I/O port instructions.

Basic Input and Output Interfaces

The basic input device is a set of three-state buffers. The basic output device is a set of data latches. The term IN refers to moving data from the I/O device into the microprocessor and the term OUT refers to moving data out of the microprocessor to the I/O device.

The Basic Input Interface. Three-state buffers are used to construct the 8-bit input port depicted in Figure 10–3. The external TTL data (simple toggle switches in this example) are connected to the inputs of the buffers. The outputs of the buffers connect to the data bus. The exact data bus connections depend on the version of the microprocessor. For example, the 8088 has data bus connections D7–D0, the 80486 has D31–D0, and the Pentium–Pentium 4 have D63–D0. The circuit of Figure 10–3 allows the microprocessor to read the contents of the 8 switches that connect to any 8-bit section of the data bus when the select signal SEL becomes a logic 0. Thus, whenever the IN instruction executes, the contents of the switches are copied into the AL register.

When the microprocessor executes an IN instruction, the I/O port address is decoded to generate the logic 0 on SEL. A 0 placed on the output control inputs (1G and 2G) of the 74ALS244 buffer causes the data input connections (A) to be connected to the data output (Y) connections. If a logic 1 is placed on the output control inputs of the 74ALS244 buffer, the device enters the three-state high-impedance mode that effectively disconnects the switches from the data bus.

This basic input circuit is not optional and must appear any time that input data are interfaced to the microprocessor. Sometimes it appears as a discrete part of the circuit, as shown in Figure 10–3; sometimes it is built into a programmable I/O device.

16- or 32-bit data can also be interfaced to various versions of the microprocessor, but this is not nearly as common as using 8-bit data. To interface 16 bits of data, the circuit in Figure 10–3 is doubled to include two 74ALS244 buffers that connect 16 bits of input data to the 16-bit data bus. To interface 32 bits of data, the circuit is expanded by a factor of 4.

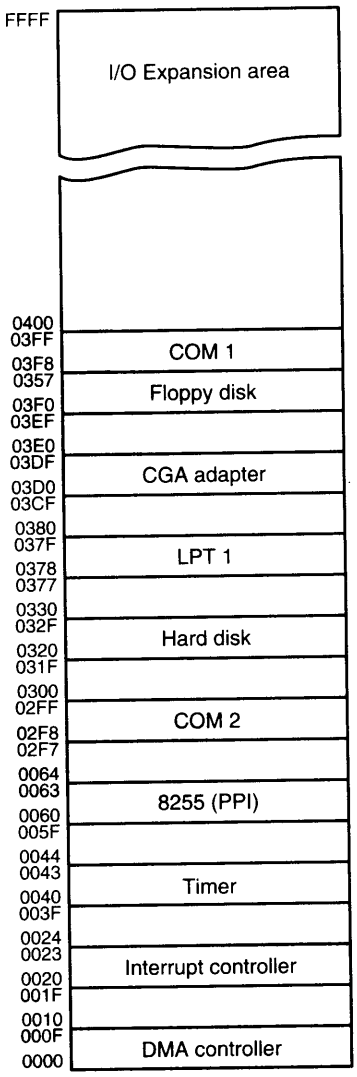


FIGURE 10–2 The I/O map of a personal computer illustrating many of the fixed I/O areas.

The Basic Output Interface. The basic output interface receives data from the microprocessor and must usually hold it for some external device. Its latches or flip-flops, like the buffers found in the input device, are often built into the I/O device.

Figure 10–4 shows how eight simple light-emitting diodes (LEDs) connect to the microprocessor through a set of eight data latches. The latch stores the number output by the microprocessor from the data bus so that the LEDs can be lit with any 8-bit binary number. Latches are needed to hold the data because when the microprocessor executes an OUT instruction, the data are only present on the data bus for less than 1.0 μs. Without a latch, the viewer would never see the LEDs illuminate.

When the OUT instruction executes, the data from AL or AX are transferred to the latch via the data bus. Here, the D inputs of a 74ALS374 octal latch are connected to the data bus to capture the output data, and the Q outputs of the latch are attached to the LEDs. When a Q output becomes a logic 0, the LED lights. Each time that the OUT instruction executes, the SEL signal to the latch activates, capturing the data output to the latch from any 8-bit section of the data bus. The data are held until the next OUT instruction executes. Thus, whenever the output instruction is executed in this circuit, the data from the AL register appear on the LEDs.

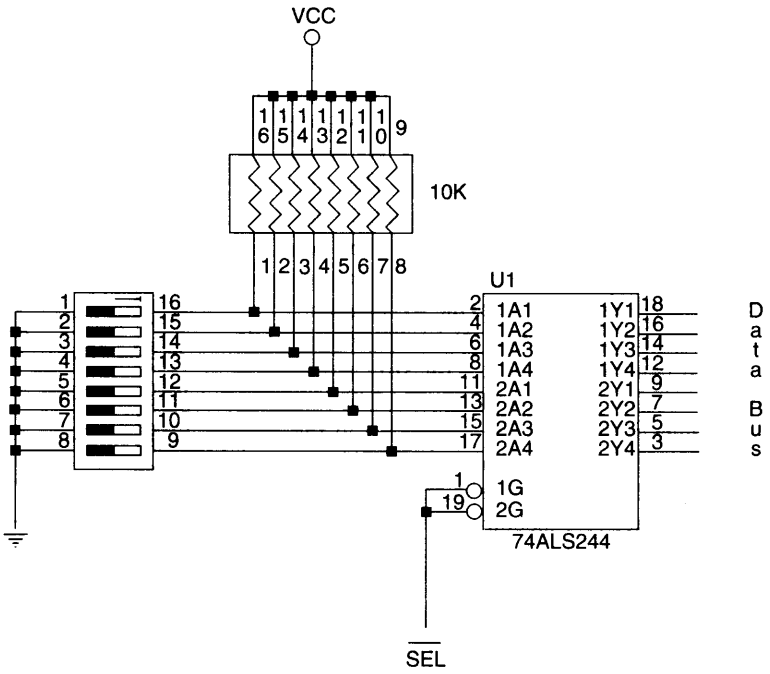


FIGURE 10–3 The basic input interface illustrating the connection of eight switches. Note that the 74ALS244 is a three-state that controls the application of the switch data to the data bus.

Handshaking

Many I/O devices accept or release information at a much slower rate than the microprocessor. Another method of I/O control, called **handshaking** or **polling,** synchronizes the I/O device with the microprocessor. An example device that requires handshaking is a parallel printer that prints 100 characters per second **(CPS).** It is obvious that the microprocessor can send more than 100 CPS to the printer, so a way to slow the microprocessor down to match speeds with the printer must be developed.

Figure 10–5 illustrates the typical input and output connections found on a printer. Here, data are transferred through a series of data connections (D7–D0), BUSY indicates that the printer is busy and STB is a clock pulse used to send data into the printer for printing.

The ASCII data to be printed by the printer are placed on D7–D0, and a pulse is then applied to the STB connection. The strobe signal sends or clocks the data into the printer so that it can be printed. As soon as the printer receives the data, it places a logic 1 on the BUSY pin, indicating that the printer is busy printing data. The microprocessor software polls or tests the BUSY pin to decide whether the printer is busy. If the printer is busy, the microprocessor waits; if it is not busy, the microprocessor sends the next ASCII character to the printer. This process of interrogating the printer is called **handshaking** or **polling.** Example 10–1 illustrates a simple procedure that tests the printer BUSY flag and then sends data to the printer if it is not busy. The PRINT procedure prints the ASCII-coded contents of BL only if the BUSY flag is a logic 0, indicating that the printer is not busy. This procedure is called each time a character is to be printed.

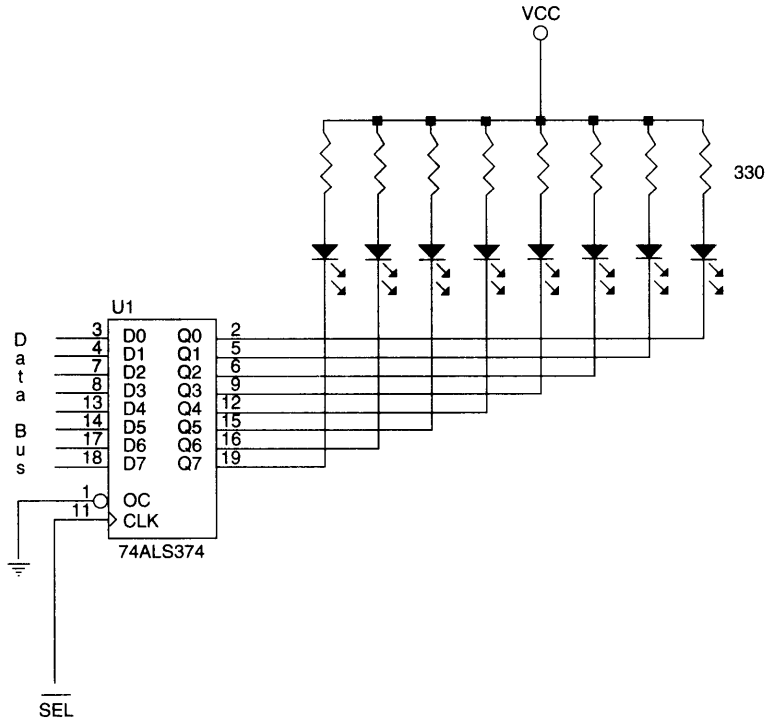


FIGURE 10–4 The basic output interface connected to a set of LED displays.

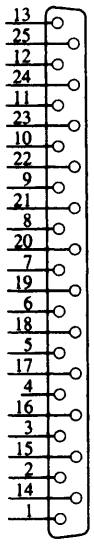
EXAMPLE 10–1

```
                        ;A procedure that prints the ASCII contents of BL.
                        ;
0000                    PRINT   PROC    NEAR

0000  E4 4B                     IN      AL,BUSY             ;get BUSY flag
0002  A8 04                     TEST    AL,BUSY_BIT         ;test BUSY bit
0004  75 FA                     JNE     PRINT               ;if printer busy
0006  8A C3                     MOV     AL,BL               ;get data from BL
0008  E6 4A                     OUT     PRINTER,AL          ;send data to printer
000A  CB                        RET                         ;return from procedure

000B                    PRINT   ENDP
```

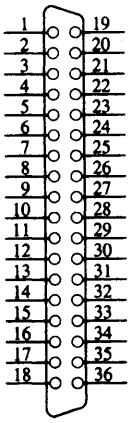
Notes About Interfacing Circuitry

A certain part of interfacing requires some knowledge about electronics. This portion of the introduction to interfacing examines some facts about electronic interfacing. Before a circuit or device can be interfaced to the microprocessor, the terminal characteristics of the microprocessor and its associated interfacing components must be known. (This was introduced at the start of Chapter 9.)

Input Devices. Input devices are already TTL and compatible, and therefore can be connected to the microprocessor and its interfacing components or they are switch-based. Most switch-based devices are either open or connected. These are not TTL levels—TTL levels are a logic 0 (0.0 V–0.8 V) or a logic 1 (2.0 V–5.0 V).



Connector DB25



Connector CENT36

DB25 Pin number	CENT36 Pin number	Function	DB25 Pin number	CENT36 Pin number	Function
1	1	$\overline{\text{Data Strobe}}$	12	12	Paper empty
2	2	Data 0 (D0)	13	13	Select
3	3	Data 1 (D1)	14	14	Afd
4	4	Data 2 (D2)	15	32	$\overline{\text{Error}}$
5	5	Data 3 (D3)	16	—	$\overline{\text{RESET}}$
6	6	Data 4 (D4)	17	31	Select in
7	7	Data 5 (D5)	18—25	19—30	Ground
8	8	Data 6 (D6)	—	17	Frame groun
9	9	Data 7 (D7)	—	16	Ground
10	10	$\overline{\text{Ack}}$	—	33	Ground
11	11	Busy			

FIGURE 10–5 The DB25 connector found on computers and the Centronics 36-pin connector found on printers for the Centronics parallel printer interface.

For a switch-based device to be used as a TTL-compatible input device, some conditioning must be applied. Figure 10–6 shows a simple toggle switch that is properly connected to function as an input device. Notice that a pull-up resistor is used to ensure that when the switch is open, the output signal is a logic 1; when the switch is closed, it connects to ground, producing a valid logic 0 level. The value of the pull-up resistor is not critical—it merely assures that the signal is at a logic 1 level. A standard range of values for pull-up resistors is usually anywhere between 1K Ω to 10K ✷.

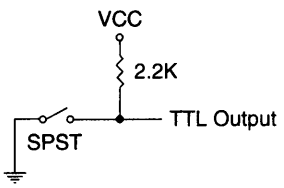


FIGURE 10–6 A single-pole, single-throw switch interfaced as a TTLdevice.

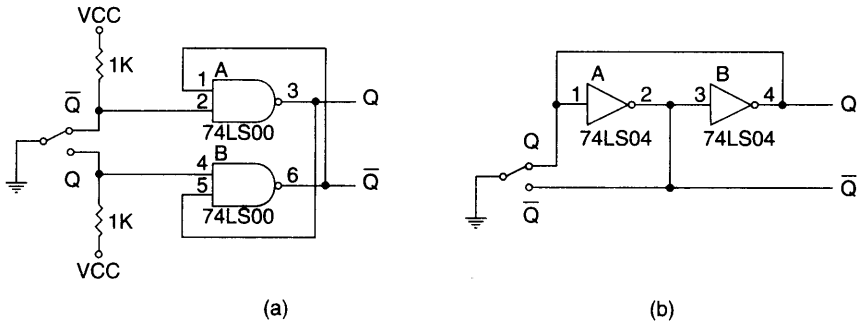


FIGURE 10–7 Debouncing switch contacts: (a) conventional debouncing and (b) practical debouncing.

Mechanical switch contacts physically bounce when they are closed, which can create a problem if a switch is used as a clocking signal for a digital circuit. To prevent problems with bounces, one of the two circuits depicted in Figure 10–7 can be constructed. The first circuit (a) is a classical textbook bounce eliminator; the second (b) is a more practical version of the same circuit. Because the first version costs more money to construct, in practice, the second would be used because it requires no pull-up resistors and only two inverters instead of two NAND gates.

You may notice that both circuits in Figure 10–7 are asynchronous flip-flops. The circuit of (b) functions in the following manner: Suppose that the switch is currently at position Q . If it is moved toward Q but does not yet touch Q, the Q output of the circuit is a logic 0. The logic 0 state is remembered by the inverters. The output of inverter B connects to the input of inverter A. Because the output of inverter B is a logic 0, the output of inverter A is a logic 1. The logic 1 output of inverter A maintains the logic 0 output of inverter B. The flip-flop remains in this state until the moving switch-contact first touches the Q connection. As soon as the Q input from the switch becomes a logic 0, it changes the state of the flip-flop. If the contact bounces back away from the Q input, the flip-flop remembers and no change occurs, thus eliminating any bounce.

Output Devices. Output devices are far more diverse than input devices, but many are interfaced in a uniform manner. Before any output device can be interfaced, we must understand what the voltages and currents are from the microprocessor or a TTL interface component. The voltages are TTL-compatible from the microprocessor of the interfacing element. (Logic 0 = 0.0 V to 0.4 V; logic 1 = 2.4 V to 5.0 V.) The currents for a microprocessor and many microprocessor-interfacing components are less than for standard TTL components. (Logic 0 = 0.0 to 2.0 mA; logic 1 = 0.0 to 400 ΩA.)

Once the output currents are known, we can now interface a device to one of the outputs. Figure 10–8 shows how to interface a simple LED to a microprocessor peripheral pin. Notice that a transistor driver is used in Figure 10–8(a) and a TTL inverter is used in Figure 10–8(b). The TTL inverter (standard version) provides up to 16 mA of current at a logic 0 level, which is more than enough to drive a standard LED. A standard LED requires 10 mA of forward bias current to light. In both circuits, we assume that the voltage drop across the LED is about 2.0 V. The data sheet for an LED states that the nominal drop is 1.65 V, but we know from experience that the drop is anywhere between 1.5 V and 2.0 V. This means that the value of the current-limiting resistor is $^{3.0\ V}/_{10}$ mA or 300 ✷. Since 300 ✷ is not a standard resistor value, we choose to use a 330 ✷ resistor.

In the circuit of Figure 10–8(a), we elected to use a switching transistor in place of the TTL buffer. The 2N2222 is a good general-purpose switching transistor that has a minimum gain of 100. In this circuit, the collector current is 10 mA, so the base current will be 1/100 of the collector current of 0.1 mA. To determine the value of the base current-limiting resistor, we use the 0.1 mA current and a voltage drop of 1.7 V. The TTL input signal has a minimum value of 2.4 V and the drop across the emitter-base junction is 0.7 V. The difference is 1.7

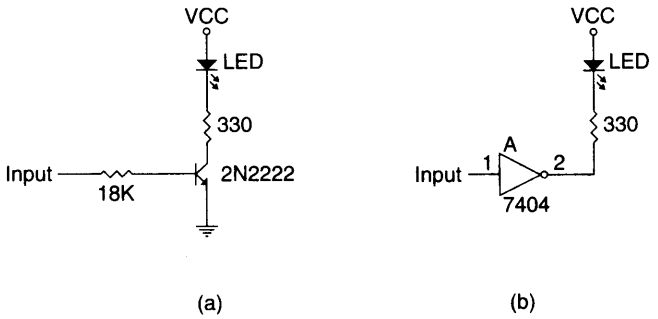


FIGURE 10–8 Interfacing an LED: (a) using a transistor and (b) using an inverter.

V, which is the voltage drop across the resistor. The value of the resistor is $^{1.7\ V}/0.1$ mA or 17K ✷. Because 17K ✷ is not a standard value, we used an 18K ✷ resistor.

Suppose that we need to interface a 12 V DC motor to the microprocessor and the motor current is 1A. Obviously, we cannot use a TTL inverter for two reasons: the 12 V signal would burn out the inverter and the amount of current far exceeds the 16 mA maximum current from the inverter. We cannot use a 2N2222 transistor either, because the maximum amount of current is 250 mA to 500 mA, depending on the package style. The solution is to use a Darlington-pair.

Figure 10–9 illustrates the motor connected to the Darlington-pair. The Darlington-pair has a minimum current gain of 7000 and a maximum current of 4A. The value of the bias resistor is calculated like the one used in the LED driver. The current through the resistor is 1A/7000, or about 0.143 mA. The voltage drop across the resistor is 0.9 V because we have two diode drops instead of one. The value of the bias resistor is $^{0.9\ V}/0.143$ mA or 6.29K✷. The standard value of 6.2 K✷ is used in the circuit. The Darlington-pair must be heat-sinked because of the amount of current going through it and the diode must also be present to prevent the Darlington-pair from being destroyed by the inductive kick-back from the motor. This circuit is also used to interface mechanical relays or just about any device that requires a large amount of current or a change in voltage.

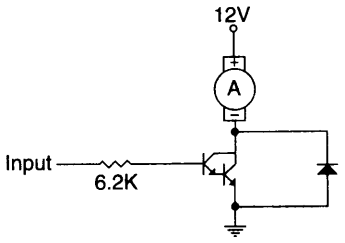


FIGURE 10–9 A DC motor interfaced to a system by using a Darlington-pair.

10–2 I/O PORT ADDRESS DECODING

I/O port address decoding is very similar to memory address decoding, especially for memory-mapped I/O devices. In fact, we do not discuss memory-mapped I/O decoding because it is treated the same as memory (except that the IORC and IOWC are not used because there is no IN or OUT instruction). The decision to use memory-mapped I/O is often determined by the size of the memory system and the placement of the I/O devices in the system.

The main difference between memory decoding and isolated I/O decoding is the number of address pins connected to the decoder. We decode A31–A0, A23–A0, or A19–A0 for memory; and A15–A0 for isolated I/O. Sometimes, if the I/O devices use only fixed I/O addressing, we decode only A7–A0. In the personal computer system, we always decode all 16 I/O port address bits. Another difference is that we use the IORC and

IOWC to activate I/O devices for a read or write operation. On earlier versions of the microprocessor, IO/M = 1 and RD or WR are used to activate I/O devices. On the newest versions of the microprocessor, the M/IO = 0 and W/R are used to activate I/O devices.

Decoding 8-Bit I/O Addresses

As mentioned, the fixed I/O instruction uses an 8-bit I/O port address that appears on A15–A0 as 0000H–00FFH. If a system will never contain more than 256 I/O devices, we often decode only address connections A7–A0 for an 8-bit I/O port address. Thus, we ignore address connection A15–A8. Embedded systems often use 8-bit port addresses. Please note that the DX register can also address I/O ports 00H–FFH. If the address is decoded as an 8-bit address, we can never include I/O devices that use a 16-bit I/O address.

Figure 10–10 illustrates a 74ALS138 decoder that decodes 8-bit I/O ports F0H through F7H. (We assume that this system will only use I/O ports 00H–FFH for this decoder example.) This decoder is identical to a memory address decoder except we only connect address bits A7–A0 to the inputs of the decoder. Figure 10–11 shows the PLD version, using a PAL for this decoder. The PAL is a better decoder circuit because the number of integrated circuits has been reduced to one device. The program for the PAL appears in Example 10–2.

EXAMPLE 10–2

```
CHIP      DECODER8 PAL16L8
;pins  1   2   3   4   5   6   7   8   9   10
       A0  A1  A2  A3  A4  A5  A6  A7  NC  GND

;pins 11  12  13  14  15  16  17  18  19   20
       NC  F7  F6  F5  F4  F3  F2  F1  F0  VCC

EQUATIONS

/F0 = A7 * A6 * A5 * A4 * A3 * /A2 * /A1 * /A0
/F1 = A7 * A6 * A5 * A4 * A3 * /A2 * /A1 * A0
/F2 = A7 * A6 * A5 * A4 * A3 * /A2 * A1 * /A0
/F3 = A7 * A6 * A5 * A4 * A3 * /A2 * A1 * A0
/F4 = A7 * A6 * A5 * A4 * A3 * A2 * /A1 * /A0
/F5 = A7 * A6 * A5 * A4 * A3 * A2 * /A1 * A0
/F6 = A7 * A6 * A5 * A4 * A3 * A2 * A1 * /A0
/F7 = A7 * A6 * A5 * A4 * A3 * A2 * A1 * A0
```

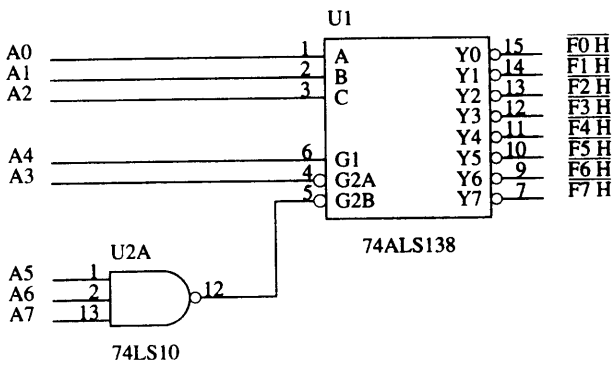


FIGURE 10–10 A port decoder that decodes 8-bit I/O ports. This decoder generates active low outputs for ports F0H–F7H.

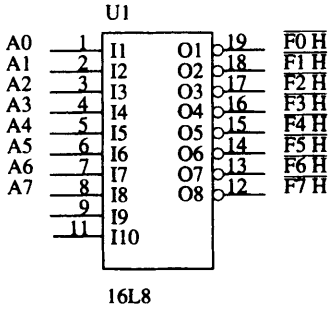


FIGURE 10–11 A PAL16L8 decoder that generates I/O port signals for port F0H–F7H.

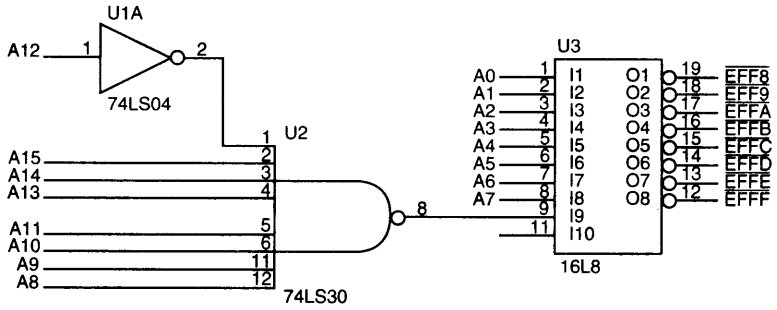


FIGURE 10–12 A PAL16L8 decoder that decodes 16-bit address EFF8H–EFFFH.

Decoding 16-Bit I/O Addresses

We also decode 16-bit I/O addresses, especially in a personal computer system. The main difference between decoding an 8-bit I/O address and a 16-bit I/O address is that eight additional address lines (A15–A8) must be decoded. Figure 10–12 illustrates a circuit that contains a PAL16L8 and an 8-input NAND gate used to decode I/O ports EFF8H–EFFFH.

The NAND gate decodes the first eight bits of the I/O port address (A15–A8). The output of the NAND gate generates a signal to enable the PAL16L8 for any I/O address between EF00H and EFFFH. The PAL16L8 further decodes the I/O address to produce eight active low output strobes EFF8H–EFFFH The program for the PAL16L8 decoder appears in Example 10–3.

EXAMPLE 10–3

```
CHIP      DECODER9 PAL16L8

;pins  1  2  3  4  5  6  7  8  9     10
       A0 A1 A2 A3 A4 A5 A6 A7 NAND GND

;pins 11     12     13     14     15     16     17     18     19  20
       NC EFFFH EFFEH EFFDH EFFCH EFFBH EFFAH EFF9H EFF8H VCC

EQUATIONS

/EFF8H = A7 * A6 * A5 * A4 * A3 * /A2 * /A1 * /A0 * /NAND
/EFF9H = A7 * A6 * A5 * A4 * A3 * /A2 * /A1 * A0 * /NAND
/EFFAH = A7 * A6 * A5 * A4 * A3 * /A2 * A1 * /A0 * /NAND
/EFFBH = A7 * A6 * A5 * A4 * A3 * /A2 * A1 * A0 * /NAND
/EFFCH = A7 * A6 * A5 * A4 * A3 * A2 * /A1 * /A0 * /NAND
/EFFDH = A7 * A6 * A5 * A4 * A3 * A2 * /A1 * A0 * /NAND
/EFFEH = A7 * A6 * A5 * A4 * A3 * A2 * A1 * /A0 * /NAND
/EFFF7 = A7 * A6 * A5 * A4 * A3 * A2 * A1 * A0 * /NAND
```

8- And 16-Bit I/O Ports

Now that we understand that decoding the I/O port address is probably simpler than decoding a memory address (because of the number of bits), we explain how data are transferred between the microprocessor and 8- or 16-bit I/O devices. Data transferred to an 8-bit I/O device exist in one of the I/O banks in a 16-bit microprocessor such

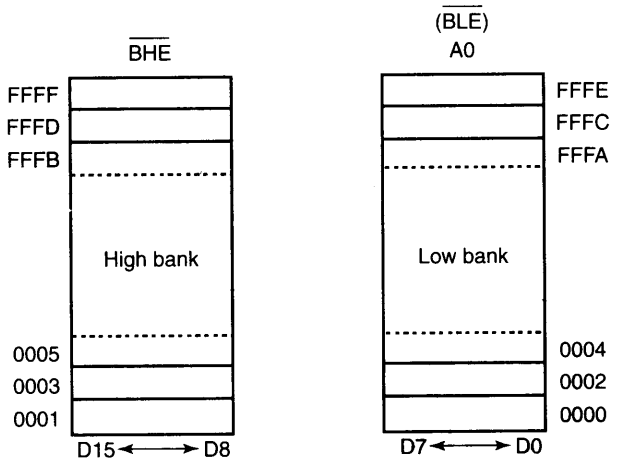


FIGURE 10—13 The I/O banks found in the 8086, 80186, 80286, and 80386SX.

as the 80386SX. The I/O system contains two 8-bit memory banks, just as memory does. This is illustrated in Figure 10–13, which shows the separate I/O banks for a 16-bit system such as the 80386SX.

Because two I/O banks exist, any 8-bit I/O write requires a separate write strobe to function correctly. I/O reads do not require separate read strobes. As with memory, the microprocessor reads only the byte it expects and ignores the other byte. The only time that a read can cause problems is when the I/O device responds incorrectly to a read operation. In the case of an I/O device that responds to a read from the wrong bank, we may need to include separate read signals. This is discussed if the case arises later in this chapter.

Figure 10–14 illustrates a system that contains two different 8-bit output devices, located at 8-bit I/O address 40H and 41H. Because these are 8-bit devices and because they appear in different I/O banks, we generate separate I/O write signals. Note that all I/O ports use 8-bit addresses. Thus, ports 40H and 41H can each be addressed as separate 8-bit ports, or together as one 16-bit port. The program for the PAL16L8 decoder used in Figure 10–14 is illustrated in Example 10–4.

EXAMPLE 10–4

```
CHIP      DECODERA PAL16L8
;pins  1      2   3   4   5   6   7   8   9   10
       BHE   IOWC A0  A1  A2  A3  A4  A5  A6  GND

;pins 11  12  13  14  15  16  17  18  19  20
       A7  NC  NC  NC  NC  NC  NC  40  41  VCC

EQUATIONS

/40 = /BLE * /IOWC * /A7 * A6 * /A5 * /A4 * /A3 * /A2 * /A1
/41 = /BHE * /IOWC * /A7 * A6 * /A5 * /A4 * /A3 * /A2 * /A1
```

When selecting 16-bit wide I/O devices, the BLE (A0) and BHE pins have no function because both I/O banks are selected together. Although 16-bit I/O devices are relatively rare, a few do exist for analog-to-digital and digit-to-analog converters, as well as for some video and disk memory interfaces.

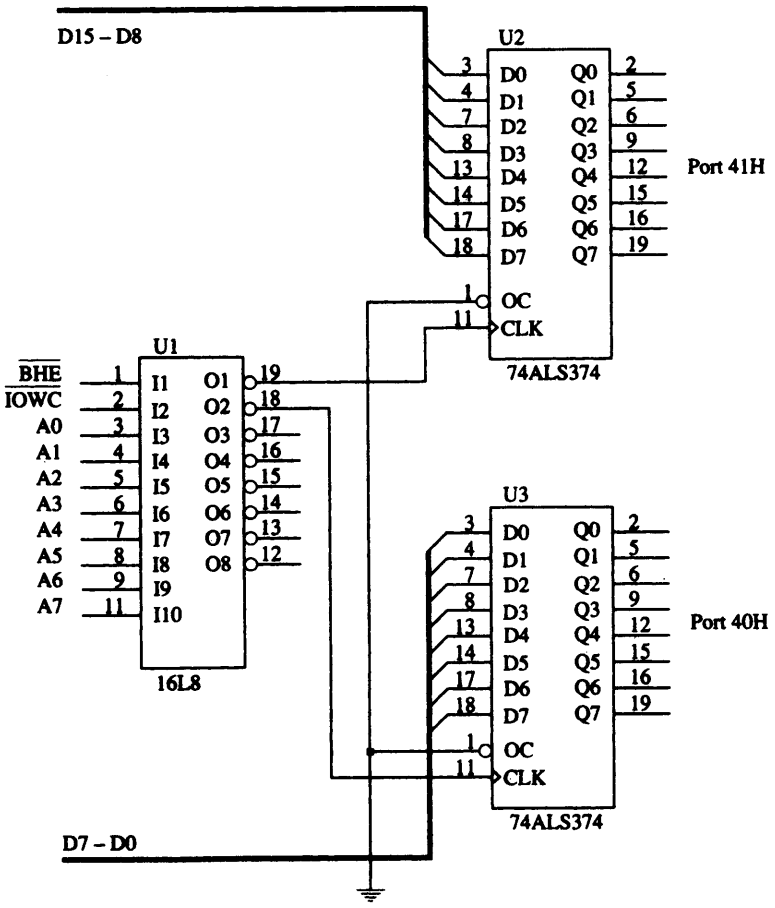


FIGURE 10–14 An I/O port decoder that selects ports 40H and 41H for output data.

Figure 10–15 illustrates a 16-bit input device connected to function at 8-bit I/O addresses 64H and 65H. Notice that the PAL16L8 decoder does not have a connection for address bit BLE (A0) and BHE because these signals do not apply to 16-bit wide I/O devices. The program for the PAL16L8, illustrated in Example 10–5, shows how the enable signals are generated for the three-state buffers (74ALS244) used as input devices.

EXAMPLE 10–5

```
CHIP      DECODERB PAL16L8

;pins    1   2   3   4   5   6   7   8   9   10
      IORC A1  A2  A3  A4  A5  A6  A7  NC  GND
;pins 11  12  13  14  15  16  17  18  19   20
       NC  NC  NC  NC  NC  NC  NC  NC  O6X VCC

EQUATIONS

/O64 = /IORC * /A7 * A6 * A5 * /A4 * /A3 * A2 * /A1
```

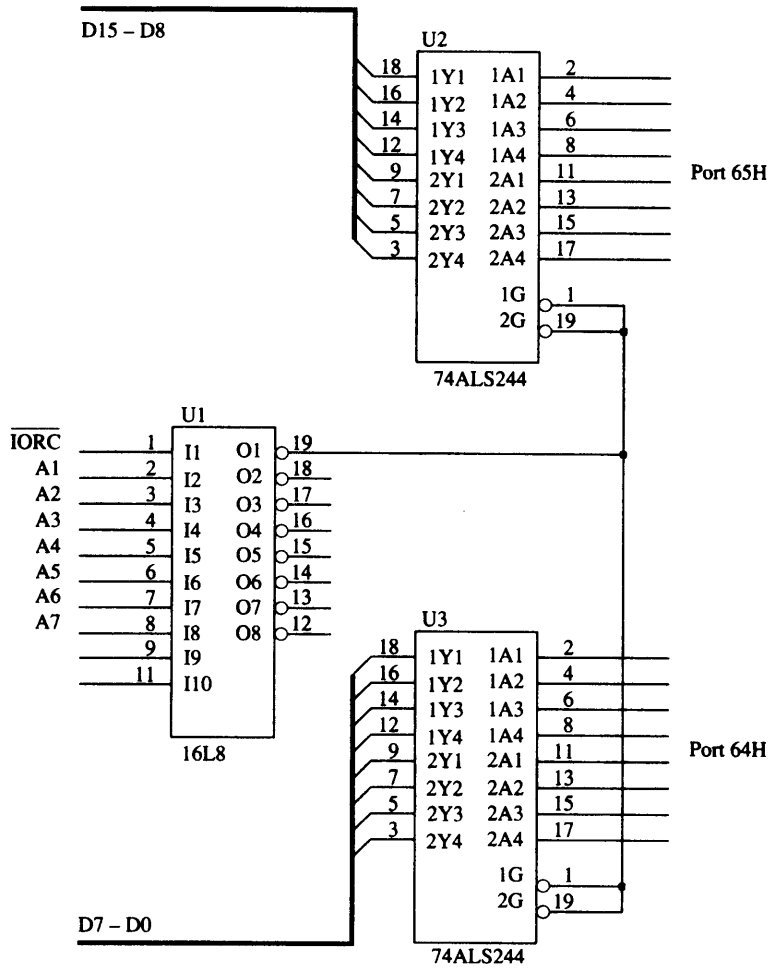


FIGURE 10–15 A 16-bit I/O port decoded at I/O addresses 64H and 65H.

32-Bit Wide I/O Ports

Although 32-bit wide I/O ports are not common, they may eventually become commonplace because of newer buses found in computer systems. The once-promising EISA system bus supports 32-bit I/O as well as the VESA local and current PCI bus, but these are found only in some IO systems.

The circuit of Figure 10–16 illustrates a 32-bit input port for the 80386DX or 80486 microprocessor. As with prior interfaces, this circuit uses a single PAL to decode the I/O ports and four 74LS244 buffers to connect the I/O data to the data bus. The I/O ports decoded by this interface are the 8-bit ports 70H–73H, as illustrated by the PAL program in Example 10–6. Again, we only decode an 8-bit I/O port address.

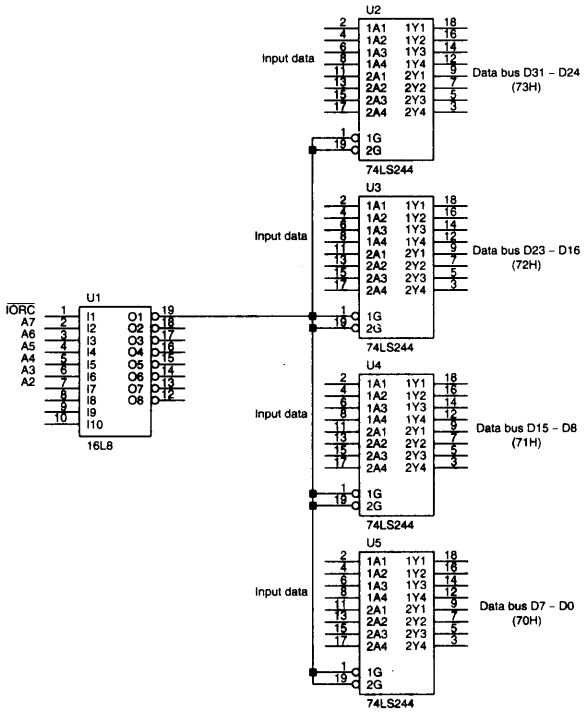


FIGURE 10–16 A 32-bit input port decoded at bytes 70H–73H for the 80386DX through the Pentium 4.

EXAMPLE 10–6

```
CHIP     DECODERC PAL16L8

;pins    1  2  3  4  5  6  7  8  9  10
      IORC A7 A6 A5 A4 A3 A2 NC NC GND

;pins 11 12 13 14 15 16 17 18 19  20
      NC NC NC NC NC NC NC NC SEL VCC

EQUATIONS

/SEL = /IORC * /A7 * A6 * A5 * A4 * /A3 * /A2
```

With the Pentium–Pentium 4 micorprocessors and their 64-bit data buses, I/O ports appear in various banks, as determined by the I/O port address. For example, 8-bit I/O port 0034H appears in Pentium I/O bank 5, while the 16-bit I/O port 0034H–0035H appears in Pentium banks 5 and 6. A 32-bit I/O access in the Pentium system can appear in any four consecutive I/O banks. For example, 32-bit I/O port 0100H–0103H appears in banks 0–3. How is a 64-bit I/O device interfaced? The widest I/O transfers are 32 bits, and currently there are no 64-bit I/O instructions to support 64-bit transfers.

10–3 THE PROGRAMMABLE PERIPHERAL INTERFACE

The 82C55 **programmable peripheral interface** (PPI) is a very popular, low-cost interfacing component found in many applications. The PPI, which has 24 pins for I/O that are programmable in groups of 12 pins, has groups that operate in three distinct modes of operation. The 82C55 can interface any TTL-compatible I/O device to the microprocessor. The 82C55 (CMOS version) requires the insertion wait states if operated with a microprocessor using higher than an 8 MHz clock. It also provides at least 2.5 mA of sink (logic 0) current at each output, with a maximum of 4.0 mA. Because I/O devices are inherently slow, wait states used during I/O transfers do not impact significantly upon the speed of the system. The 82C55 still finds application (compatible for programming, although it may not appear in the system as a discrete 82C55), even in the latest Pentium 4-based computer system. The 82C55 is used for interface to the keyboard and the parallel printer port in many personal computers, but it is found as a function within a interfacing chipset. The chipset also controls the timer and reads data from the keyboard interface.

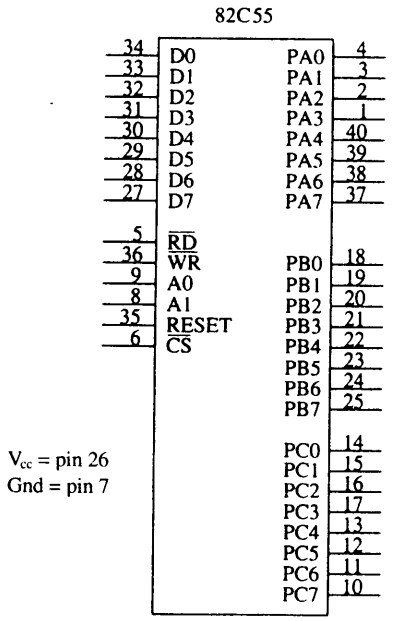


FIGURE 10–17 The pin-out of the 82C55 programmable peripheral interface.

Basic Description of the 82C55

Figure 10–17 illustrates the pin-out diagram of the 82C55. Its three I/O ports (labeled A, B, and C) are programmed as groups. Group A connections consist of port A (PA7–PA0) and the upper half of port C (PC7–PC4), and group B consists of port B (PB7–PB0) and the lower half of port C (PC3–PC0). The 82C55 is selected by its CS pin for programming and for reading or writing to a port. Register selection is accomplished through the A1 and A0 input pins that select an internal register for programming or operation. Table 10–2 shows the I/O port assignments used for programming and access to the I/O ports. In the personal computer, an 82C55 or its equivalent is decoded at I/O ports 60H–63H.

TABLE 10–2 I/O port assignments for the 82C55.

A_1	A_0	*Function*
0	0	Port A
0	1	Port B
1	0	Port C
1	1	Command Register

The 82C55 is a fairly simple device to interface to the microprocessor and program. For the 82C55 to be read or written, the CS input must be a logic 0 and the correct I/O address must be applied to the A1 and A0 pins. The remaining port address pins are don't cares as far as the 82C55 is concerned, and are externally decoded to select the 82C55.

Figure 10–18 shows an 82C55 connected to the 80386SX so that it functions at 8-bit I/O port addresses C0H (port A), C2H (port B), C4H (port C), and C6H (command register). This interface uses the low bank of the 80386SX I/O map. Notice from this interface that all the 82C55 pins are direct connections to the 80386SX, except for the CS pin. The CS pin is decoded and selected by a 74ALS138 decoder.

The RESET input to the 82C55 initializes the device whenever the microprocessor is reset. A RESET input to the 82C55 causes all ports to be set up as simple input ports using mode 0 operation. Because the port pins are internally programmed as input pins after a RESET, damage is prevented when the power is first applied to the system. After a RESET, no other commands are needed to program the 82C55, as long as it is used as an input device for all three ports. Note that an 82C55 is interfaced to the personal computer at port addresses 60H–63H for

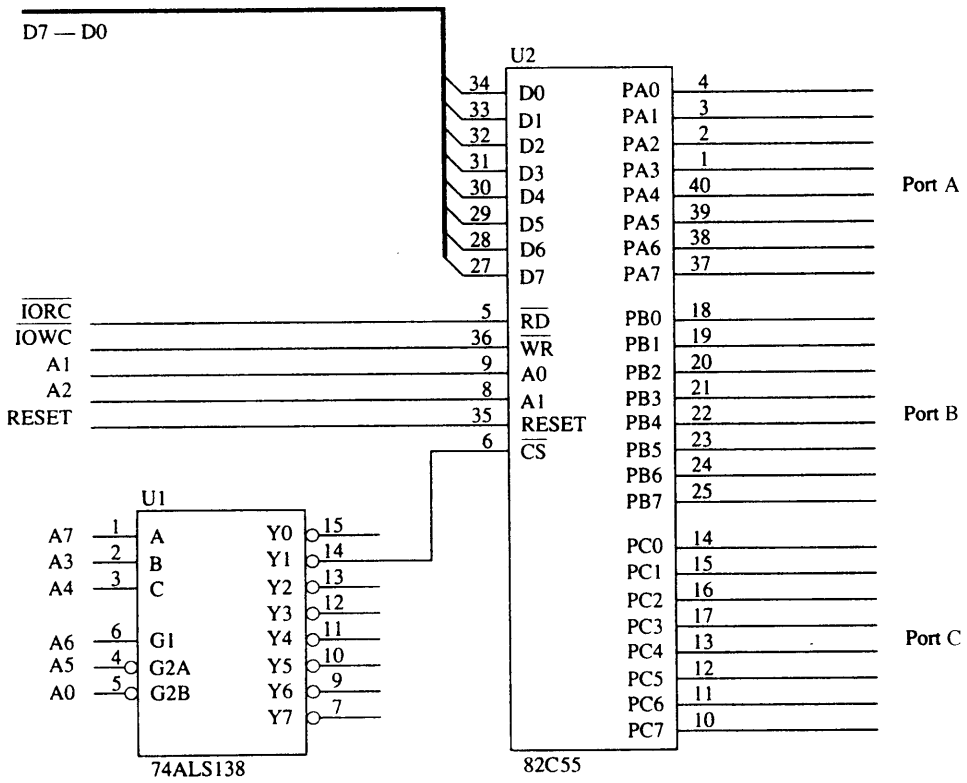


FIGURE 10–18 The 82C55 interfaced to the low bank of the 80386SX microprocessor.

keyboard control; and also for controlling the speaker, timer, and other internal devices such as memory expansion. This is true for any AT or earlier style personal computer system.

Programming the 82C55

The 82C55 is programmed through the two internal command registers that are illustrated in Figure 10–19. Notice that bit position 7 selects either command byte A or command byte B. Command byte A programs the function of group A and B, while command byte B sets (1) or resets (0) bits of port C only if the 82C55 is programmed in mode 1 or 2.

Group B pins (port B and the lower part of port C) are programmed as either input or output pins. Group B operates in either mode 0 or mode 1. Mode 0 is the basic input/output mode that allows the pins of group B to be programmed as simple input and latched output connections. Mode 1 operation is the strobed operation for group B connections, where data are transferred through port B and handshaking signals are provided by port C.

Group A pins (port A and the upper part of port C) are programmed as either input or output pins. The difference is that group A can operate in modes 0, 1, and 2. Mode 2 operation is a bi-directional mode of operation for port A.

If a 0 is placed in bit position 7 of the command byte, command byte B is selected. This command allows any bit of port C to be set (1) or reset (0), if the 82C55 is operated in either mode 1 or 2. Otherwise, this command byte is not used for programming. We often use the bit set/reset function in a control system to set or clear a control bit at port C. The bit set/reset function is glitch-free, which means that the other port C pins will not change during the bit set/reset command.